HACK X CRACK: UN SERVIDOR FTP EN LA CONSOLA XBOX --> EVOLUTION X



ESTE MES 16 páginas extra!!!

PHP: EL REY

TRABAJANDO CON FICHEROS Y DIRECTORIOS

ABRIENDO ARCHIVOS REMOTOS CSV

EN EL PROXIMO NUMERO: HACKEA WINDOWS EN 40 SEGUNDOS

NUMERO 16

LOS CUADERNOS DE HACK X CRACK

www.hackxcrack.com

PROGRAMACION DEL SISTEMA



I.P.C. COMUNICACION ENTRE PROCESOS

DESTRIPANDO LOS PROTOCOLOS DE INTERNET

XBOX



HACIENDO COPIAS DE SEGURIDAD DE TUS JUEGOS

USENET: UNA INMENSA RED DE FOROS

LA SOCIEDAD DE USENET

JERARQUIA DE NEWSGROUPS

PROTOCOLO NNTP

TOPOLOGÍA DE USENET

Y MUCHO +

3 SERVIDORES ON LINE PARA TUS PRACTICAS DE HACK

ULTIMA ENTREGA DEL CURSO XML

DOMNode

HACK X CRACK - HACK X CRACK - HACK X CRACK - HACK X CRACK

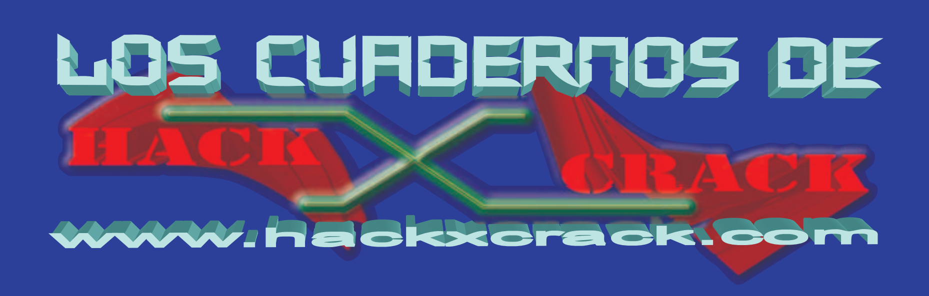
Nº 16 -- P.V.P. 4,5 EUROS



LOS MEJORES ARTÍCULOS GRATIS EN NUESTRA WEB

PC PASO A PASO: ABRIENDO ARCHIVOS REMOTOS CON PHP





EDITORIAL: EDITOTRANS S.L.
C.I.F: B43675701
PERE MARTELL Nº 20, 2º - 1ª
43001 TARRAGONA (ESPAÑA)

Director Editorial
I. SENTIS
E-mail contacto
director@editotrans.com
Título de la publicación
Los Cuadernos de HACK X CRACK.
Nombre Comercial de la publicación
PC PASO A PASO
Web: www.hackxcrack.com
Dirección: PERE MARTELL Nº 20, 2º - 1ª.
43001 TARRAGONA (ESPAÑA)



Director de la Publicación
J. Sentís

E-mail contacto
director@hackxcrack.com

Diseño gráfico:
J. M. Velasco

E-mail contacto:
grafico@hackxcrack.com

Redactores
AZIMUT, ROTEADO, FASTIC, MORDEA, FAUSTO, ENTROPIC, MEIDOR, HASHIMUIRA, BACKBONE, ZORTEMIUS, AK22, DORKAN, KMORK, MAILA, TITINA, SIMPSIM... ... ...

Contacto redactores
redactores@hackxcrack.com

Colaboradores
Mas de 130 personas: de España, de Brasil, de Argentina, de Francia, de Alemania, de Japón y algún Estadounidense.

E-mail contacto
colaboradores@hackxcrack.com

Imprime
I.G. PRINTONE S.A. Tel 91 808 50 15

DISTRIBUCIÓN:
SGEL, Avda. Valdeparra 29 (Pol. Ind.)
28018 ALCOBENDAS (MADRID)
Tel 91 657 69 00 FAX 91 657 69 28
WEB: www.sgel.es

TELÉFONO DE ATENCIÓN AL CLIENTE: 977 22 45 80
Petición de Números atrasados y Suscripciones (Srta. Genoveva)
HORARIO DE ATENCIÓN: DE 9:30 A 13:30
(LUNES A VIERNES)

NUMERO 16 -- PRINTED IN SPAIN
PERIOCIDAD MENSUAL
Deposito legal: B.26805-2002
Código EAN: 8414090202756

EDITORIAL

EL TIEMPO Y LAS PROMESAS

Ante todo una ración de disculpas. Hasta el día 21 de Enero no pudimos tener la WEB operativa al 100% y muchos lectores no pudieron bajarse los programas y artículos liberados. Lamentamos este retraso y no hay excusa que valga, podemos decir que tuvimos problemas con la forma en que el servidor interpretaba nuestra codificación en PHP (cosa que es cierta), pero eso no importa, deberíamos haber tenido La Web al 100% desde el día 1 de Enero.

Otra más… se ha recibido muchas protestas por el mal funcionamiento de los Servidores de Hack. Azimut ha estado retocando los protocolos de reinstalación semanal y esperamos que a partir de ahora se cumplan las reinstalaciones y los servidores se mantengan ON LINE. No obstante, debemos tener en cuenta que los servidores están para ser hackeados y que algunas personas los dejan inoperativos cuando hacen sus prácticas.

Otra de las grandes peticiones es que escribamos artículos de Hacking Básico. Bien, en nuestra opinión el Hacking Básico no existe (o al menos no puede llamarse hacking); pero no podemos ignorar las peticiones de nuestros lectores, así que a partir del mes que viene habrá un artículo de Hack-Básico para disfrute del personal. Empezaremos por hackear cualquier sistema Windows NT/2000/XP en 40 segundos (o menos), algo que todo "iniciado" debería saber y, sobretodo, algo que cualquier Administrador de Sistemas debería conocer.

Si crees que es imposible fulminar la seguridad de un Windows en 40 segundos, prepárate para el mes que viene!!! ;)

GRACIAS una vez más a colaboradores, lectores y foreros.

G R A C I A S

G R A C I A S

G R A C I A S

G R A C I A S

G R A C I A S

G R A C I A S

INDICE

INDICE DE ANUNCIANTES

CURSO DE PHP:

APRENDE A MANEJAR FICHEROS CON PHP

- Trabajando con ficheros y directorios desde PHP
- Abriendo ficheros remotos
- ¿Qué es CSV?

Continuamos con el curso de PHP. En este número aprenderemos a manejar tanto la lectura como la escritura. Continuaremos mejorando el programa de generación de IP, en esta ocasión las IPs en vez de ser presentadas en pantalla serán guardadas en un fichero para su posterior tratamiento. Y además aprenderemos a consultar datos de manera remota.

Manejo de ficheros y directorios.

En una aplicación creada con PHP es posible recuperar datos y guardarlos en ficheros para su posterior tratamiento. Hasta ahora todos los ejemplos que se han comentado muestran los resultados en el navegador, con lo que una vez que se cierra el navegador perdemos los datos generados. Para evitar la perdida de información podemos guardar dichos datos en un archivo para que posteriormente podamos tratarlos. ¿No sería mejor que las IPs generadas de en el ejemplo del número anterior fueran guardadas en un fichero?, pues eso será lo que haremos.

PHP ofrece al programador un conjunto de funciones y procedimientos mediante los cuales es posible acceder a un fichero de texto para leerlo, modificarlo, añadirle o quitarle líneas. Del mismo modo que podemos tratar ficheros, también podemos manejar directorios, podemos crear, renombrar y eliminar carpetas.

Manejo de ficheros

El proceso a seguir para acceder a un fichero, sea para consultar el contenido o para modificarlo va asociado a 3 pasos:

1. Abrir el fichero, para ello hay que especificar como se va a abrir.

2. La acción sobre el fichero, (escribir o leer).

3. Y por último, cerrar el fichero.

Abrir ficheros

Para abrir un fichero se utiliza la función **fpen()**. Esta función es muy potente y no tan conocida, pues permite abrir un fichero del servidor vía HTTP o FTP.

Esto último lo veremos más adelante y verás la importancia que tiene (cuando supe que **fopen()** era capaz de conectarse remotamente a otras máquinas yo alucinaba).

La sintasix es la siguiente:

fopen (string nombrefichero, string modoApertura);

La cadena **nombreFichero** debe contener la ruta completa para abrir el fichero.

El **modoApertura** especifica como se debe abrir el fichero, va asociado al tipo de acción que se desea realizar (escribir,leer).

Si **nombreFichero** comienza con "*http://*" (no es sensible a mayúsculas), se abre una conexión HTTP 1.0 hacia el servidor especificado y se devuelve un apuntador de fichero al comienzo del texto de respuesta.

No maneja redirecciones HTTP, por eso se debe incluir una barra final cuando se trata de directorios.

Si **nombreFichero** comienza con "*ftp://*" (no es sensible a mayúsculas), se abre una conexión ftp hacia el servidor especificado y se devuelve un apuntador al fichero requerido. Si el servidor no soporta ftp en modo pasivo, esto fallará (si no sabes lo que es FTP en modo pasivo descárgate gratis el número 1 de esta revista desde la Web www.hackxcrack.com).

Se pueden abrir fichero via ftp para leer o para escribir (pero no ambas cosas simultáneamente).

Existen seis modalidades diferentes para abrir un fichero:

Valor	Descripción
a	Abre sólo para escribir (añadir); sitúa el apuntador del fichero al final del mismo. Si el fichero no existe, trata de crearlo.
a+	Abre para lectura y escritura (añadiendo); sitúa el apuntador del fichero al final del mismo. Si el fichero no existe, trata de crearlo.
r	Abre para sólo lectura; sitúa el apuntador del fichero al comienzo del mismo.
r+	Abre para lectura y escritura; situa el apuntador del fichero al comienzo del fichero.
w	Abre para sólo escritura; sitúa el apuntador del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
w+	Abre el fichero para lectura y escritura; sitúa el apuntador del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
b	Esto es útil para sistemas que diferencian entre ficheros binarios y de texto (ej. es inútil en Unix). Si no se necesita, será ignorado.

El último valor de la tabla será ignorado en sistemas Unix/Linux ya que no existe distinción entre este tipo de fichero por lo que el valor será ignorado.

Primer ejemplo:
Vamos a comenzar con un sencillo ejemplo. Ejecuta el siguiente código, el cual abre el fichero php.ini, situado en la ruta c:\windows\php.ini, con operación de lectura (valor: r).



¿Como?...

¿Cómo? ¿no sabes ejecutar el código?
Ufffff, eso es que no has seguido nuestro curso de PHP. Tranquilo, puedes pedir los números atrasados de PC PASO A PASO desde nuestra Web (www.hackxcrack.com) y/o descargarte de forma gratuita los artículos liberados.

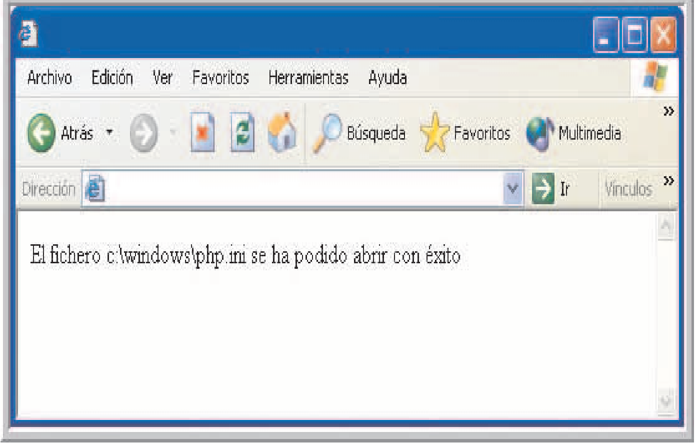
```
<?
$fichero="c:\\windows\\php.ini";    // fichero que deseamos abrir.
if (fopen($fichero,"r")) {    // comprobamos si el fichero se puede abrir para lectura.
      print ("El fichero $fichero se ha podido abrir con éxito");
} else {
      print ("Error al abrir el fichero.");
}
?>
```

En la plataforma Windows, ten cuidado de escribir correctamente las barras invertidas en el path del fichero (poniéndolas dobles), o usa barras directas.

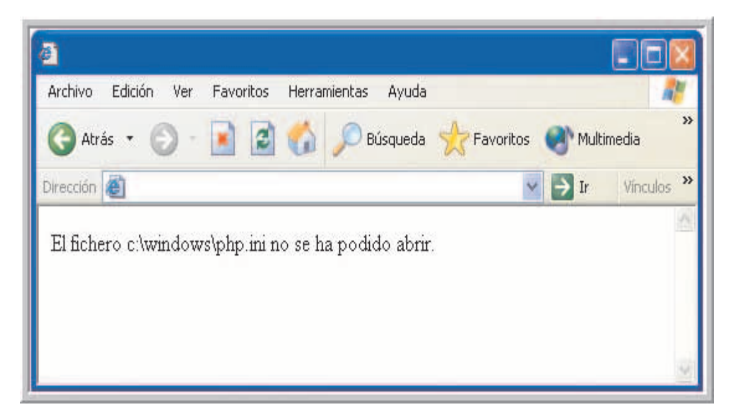
También se podría haber colocado el fopen de la siguiente manera con el mismo efecto:

```
if (fopen("c:\\windows\\php.ini","r")) {
```

Prueba a cambiar el valor **$fichero**, para que puedas comprobar ambos resultados. Experimenta con las barras inclinadas, por norma se suele utilizar las barras dobles. En el ejemplo damos por supuesto que el fichero existe, en caso de que el fichero no exista dará error de PHP.



Resultado de abrir el fichero "php.ini" con la función fopen()



Resultado al abrir el fichero con un error

¿Cuándo puede dar error al abrir un fichero?, si el fichero está abierto por otra aplicación puede que tenga todos los privilegios y no permita abrir el fichero a otros programas, por lo que tendremos que esperar a que la aplicación deje de utilizar el archivo.

En el ejemplo se abre el fichero php.ini para su lectura tal y como indica el valor "r".

Si se consigue abrir la función fopen devolverá el valor TRUE y FALSE en caso de que no haya sido posible abrir el archivo para lectura.

La función fopen() devuelve un identificador creado al abrir el fichero y que será necesario para utilizar las funciones de escribir y leer el fichero.

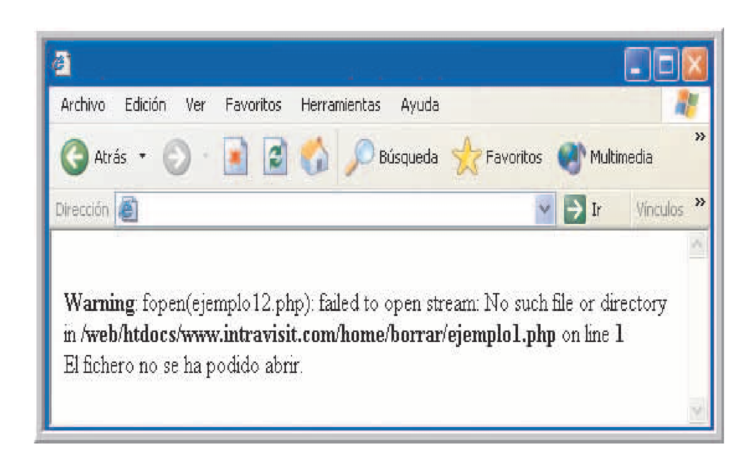
Para capturar el identificador (también llamado hadler) se podría haber utilizado la siguiente sintaxis:

```
$fi=fopen($fichero,"r");     // la variable $fi
// tomará el valor que hace referencia al fichero
```

Con **fopen()** también podemos utilizar la función **die()**. Envía un mensaje y finaliza la ejecución del script.

```
<?
$fichero="c:\\windows\\phppp.ini";
// Introducimos un error en el nombre del fichero
fopen($fichero,"r") or die ("El fichero no se ha podido abrir.");
?>
```

Con este código tenemos el siguiente resultado:



Como se puede ver en la imagen el fichero no se ha encontrado y aparece el mensaje "El fichero no se ha podido abrir", pero sigue apareciendo un mensaje de WARNING, por lo que seguimos mostrando una alerta de PHP y lógicamente no deseamos que se vea el WARNING, ¿qué se puede hacer?.

Simplemente añadiendo el simbolo @ a fopen, ¿cómo?, pues como indica el siguiente ejemplo:

```
<?
$fichero="c:\\windows\\phppp.ini"; //Introducimos un error en el nombre del fichero.
@fopen($fichero,"r") or die ("El fichero no se ha podido abrir.");
?>
```

El símbolo puede añadirse a cualquier función para evitar que se vean los WARNING, lógicamente existen soluciones mejores, lo veremos más adelante.

Cerrar ficheros

Una vez que han finalizado las tareas con el fichero es recomendable cerrar el archivo. Para ello, se utiliza la función **fclose()**.

```
<?
$fichero1=fopen($fichero1,"r");
$fichero2=fopen($fichero2,"r");
// Aquí colocamos código para tratar los archivos
//(leer, modificarlos, ...)
fclose ($fichero1); // cerramos el fichero1
fclose ($fichero2); // cerramos el fichero2
?>
```

Al finalizar el script, PHP cierra todos los ficheros abiertos que no han sido cerrados; pero esto es una mala práctica ya que durante la ejecución del código es posible que se corrompan los ficheros con la perdida de datos, es por ello que siempre se recomienda cerrar los ficheros una vez que ya no se van a utilizar.

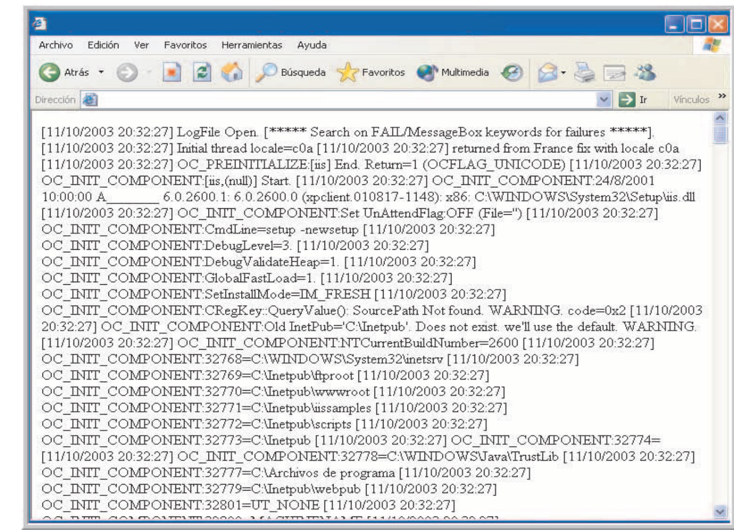
Al igual que la función **fopen()** si el proceso de cerrar el fichero se ha ejecutado correctamente devolverá el valor TRUE, de no ser así, devolverá el valor FALSE.

Mostrar el contenido de un fichero

Una vez abierto el fichero ya podemos operar con él, el siguiente ejemplo es un código que muestra el contenido de un fichero, para ello se utiliza la función fpassthru(), utilizando como argumento el identificador del fichero a abrir.

```
<?
$fichero="c:\\windows\\iis6.log";     // fichero en formato texto que se va a abrir.
if ($fp=fopen($fichero,"r")) {         // abre el fichero y crea un identificador.
fpassthru($fp);                        // muestra el contenido del fichero.
} else {
        print ("No se ha podido abrir el fichero");
}
?>
```

Observa que en **fpassrhru()** no se ha colocado el nombre del fichero, simplemente se ha indicado el identificador $fp que indica que el fichero está abierto y con permisos de lectura ("r").



Recuerda: la función **fpassthru()** muestra TODO el contenido de un fichero.

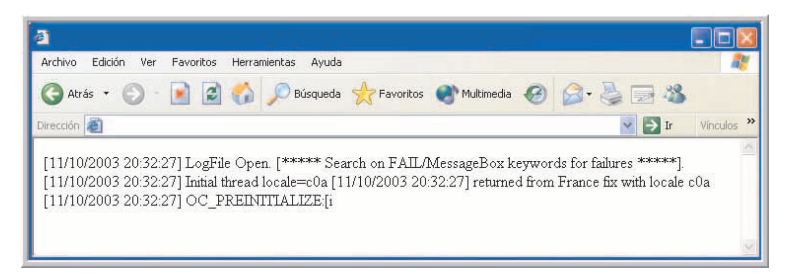
Leer desde un fichero

La mayoría de las veces solo nos interesará leer una determinada parte del fichero y no todo como en el ejemplo anterior. PHP ofrece diferentes funciones para leer determinadas partes de un fichero y la elección de la función a utilizar dependerá del dato a leer.

Para leer una cadena de un fichero abierto, se utiliza la función fread(), que necesita dos argumentos, el identificador del fichero abierto y la cantidad de caracteres a leer. Si se alcanza el final de fichero antes de completar la cantidad de caracteres indicados, solo devolverá hasta ese punto.

Por ejemplo, vamos a mostrar los primeros 255 del fichero iis6.log.

```
<?
$fichero="c:\\windows\\iis6.log";   //fichero en formato texto que se va a abrir.
if ($fp=fopen($fichero,"r")) {       // abre el fichero y crea un identificador.
      $texto=fread($fp,255);         // leemos los primeros 255 caracteres.
      print $texto;                  // mostramos en pantalla los 255 caracteres.
} else {
          print ("No se ha podido abrir el fichero");
}
?>
```





Si en tu PC...

Si en tu PC no tienes el fichero iis6.log en c:\windows\, pues créalo tu mismo con el Block de Notas (o cualquier otro editor de texto).

Otra función para leer el contenido de un fichero es **fgetc()**. A diferencia de la función anterior esta lee carácter a carácter. Por ejemplo:

```
<?
$fichero="c:\\windows\\iis6.log";      //fichero en formato texto que se va a abrir.
if ($fp=fopen($fichero,"r")) {          // abre el fichero y crea un identificador.
$caracter=fgets($fp,1);                 // lee carácter a carácter.
            print $carácter;
} else {
            print ("No se ha podido abrir el fichero");
}
?>
```

En este caso solo mostrará un carácter como resultado, aunque también nos ofrece la posibilidad de indicarle que lea una cadena indicándole así la longitud.

```
<?
$fichero="c:\\windows\\iis6.log";      // fichero en formato texto que se va a abrir.
if ($fp=fopen($fichero,"r")) {          // abre el fichero y crea un identificador.
            $caracter=fgets($fp,245);  // leemos 245 caracteres.
            print $carácter;
} else {
            print ("No se ha podido abrir el fichero");
}
?>
```

La lectura de esta cadena finalizará si el carácter de retorno de carro es encontrado o si se llega al final del fichero. De esta forma podemos abrir un archivo y leer línea a línea. ¿cómo?, muy sencillo, en el siguiente código se leerán 10 líneas del fichero y se mostrarán, fíjate que en fgets() no hemos colocado el número de caracteres a leer.

```
<?
$fichero="c:\\windows\\iis6.log";     //fichero en formato texto que se va a abrir
if ($fp=fopen($fichero,"r")) {         // abre el fichero y crea un identificador
   for ($cuenta=1;$cuenta<=10;$cuenta=$cuenta+1) {
        $caracter=fgets($fp);                 // leemos una linea del archivo
        print "Linea $cuenta del fichero: ".$caracter."<br>"; // muestra la linea del arhivo
   }
} else {
        print ("No se ha podido abrir el fichero");
}
?>
```

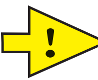
Archivo Edición Ver Favoritos Herramientas Ayuda

Linea 1 del fichero: [11/10/2003 20:32:27] LogFile Open. [***** Search on FAIL/MessageBox keywords for failures *****]
Linea 2 del fichero: [11/10/2003 20:32:27] Initial thread locale=c0a
Linea 3 del fichero: [11/10/2003 20:32:27] returned from France fix with locale c0a
Linea 4 del fichero: [11/10/2003 20:32:27] OC_PREINITIALIZE:[iis] End. Return=1 (OCFLAG_UNICODE)
Linea 5 del fichero: [11/10/2003 20:32:27] OC_INIT_COMPONENT:[iis,(null)] Start.
Linea 6 del fichero: [11/10/2003 20:32:27] OC_INIT_COMPONENT:24/8/2001 10:00:00 A______ 6.0.2600.1: 6.0.2600.0 (xpclient.010817-1148): x86: C:\WINDOWS\System32\Setup\iis.dll
Linea 7 del fichero: [11/10/2003 20:32:27] OC_INIT_COMPONENT:Set UnAttendFlag:OFF (File=")
Linea 8 del fichero: [11/10/2003 20:32:27] OC_INIT_COMPONENT:CmdLine=setup -newsetup
Linea 9 del fichero: [11/10/2003 20:32:27] OC_INIT_COMPONENT:DebugLevel=3.
Linea 10 del fichero: [11/10/2003 20:32:27] OC_INIT_COMPONENT:DebugValidateHeap=1.

Otra función similar a fgets() es fgetss(), la diferencia es que esta evita leer etiquetas de lenguajes HTML y PHP.

Es una función muy útil si se desea analizar los textos de una página HTML.

Hagamos el siguiente ejemplo, vamos a extraer los textos de la página principal de www.terra.es, para ello primero copiamos la página index en nuestra unidad local, ya sabes, ver código fuente y lo copias en tu disco duro. ¿ok?.

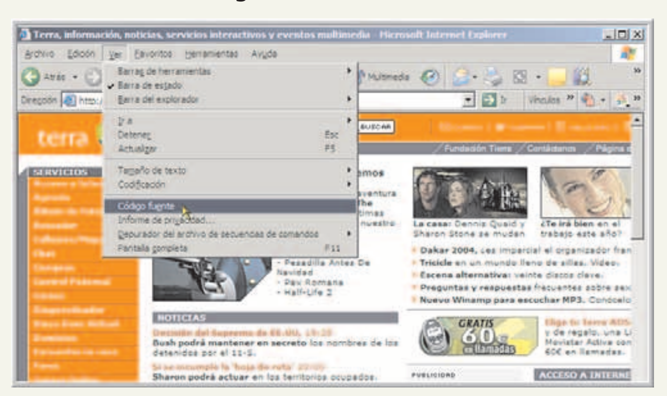


Copiando la página...

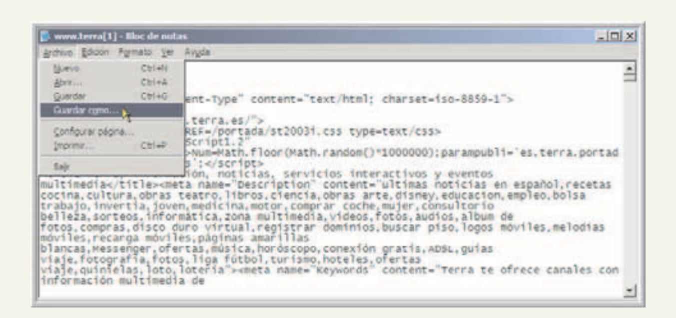
Copiando la página Web de TERRA en nuestra unidad local…

1.- Abrimos nuestro navegador (por ejemplo el Netscape o el Internet Explorer) y vamos a www.terra.es.

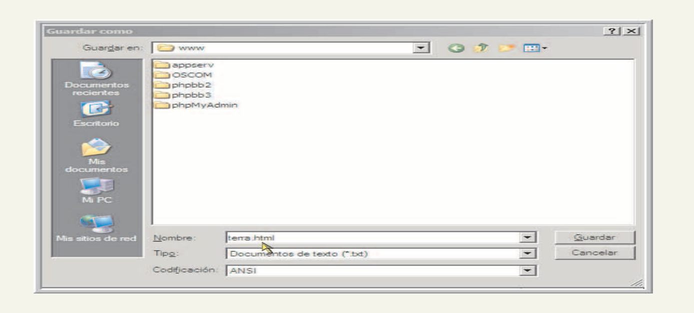
2.- Si lo hacemos con el Internet Explorer vamos al menú Ver --> Código Fuente



3.- Nos aparecerá el código en el Block de Notas. Vamos al menu Archivo --> Guardar Como



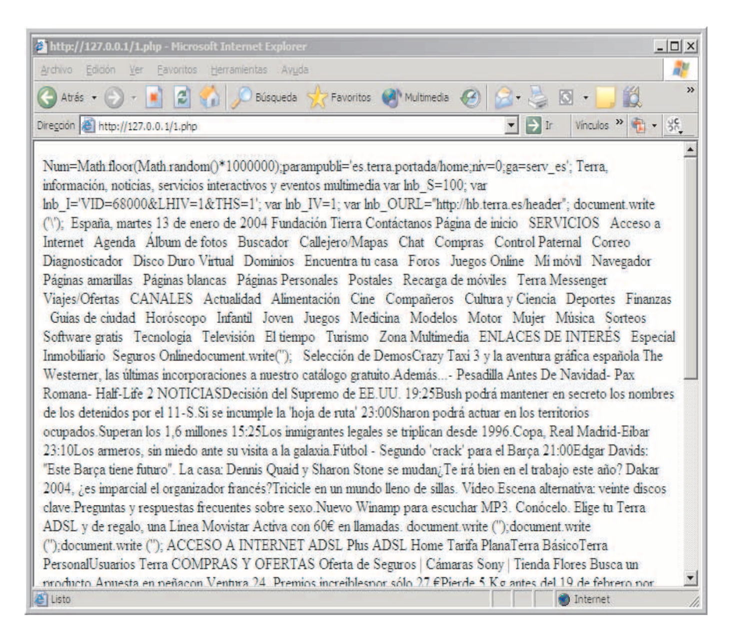
Y lo guardamos como terra.html en la ruta por defecto de nuestro Servidor Web APACHE (en nuestro caso C:\AppServ\www\)



4.- Comentario: En el código que ejecutaremos a continuación, puedes ver que no le especificamos una ruta de acceso al archivo terra.html; por lo tanto, se buscará el archivo terra.html en la ruta por defecto del Servidor APACHE, en nuestro caso C:\AppServ\www\

Venga, ejecutamos el siguiente código:

```
<?
$fichero="terra.html";          // página principal de Terra.
if ($fp=fopen($fichero,"r")) {    // abre el fichero y crea un identificador.
   for ($cuenta=1;$cuenta<=10000;$cuenta=$cuenta+1) {
   //leemos 10000 lineas si las tiene
   $caracter=fgetss($fp,64000);
   // leemos 64 Kbytes de cada linea y quitamos las etiquetas HTML
   print $caracter;             // mostramos las cadenas sin HTML
}
} else {
   print ("No se ha podido abrir el fichero");
}
?>
```



Y otra función más para leer ficheros, file(), esta función es similar a las anteriores pero guarda cada línea leída en una posición dentro de un array, siendo la primera línea el elemento cero del array.

```
<?
$fichero="terra.html";              // página principal de Terra.
if ($fp=fopen($fichero,"r")) {      // abre el fichero y crea un identificador.
$lineas=file($fp);
// lineas es un array, en cada posición se encuentra una linea del fichero leído.
for ($cuenta=0;$cuenta<count($lineas);$cuenta=$cuenta+1) {
    print ("Linea ".($cuenta+1).":".$lineas[$cuenta]."<br>");
    // Mostramos el contenido del array.
}
} else {
print ("No se ha podido abrir el fichero");
}
?>
```

En este pequeño programa hemos utilizado la función count(), sirve para averiguar el número de elementos de un array. Utilizando count() podemos conocer el número de elementos de la variable $lineas y así poder mostrar cada uno de estos. En el número anterior ya se explicó esta función.

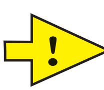
Y por último la todopoderosa función fgetcsv(), esta función es similar a fgets() excepto que fgetcsv() analiza la línea buscando campos en formato CSV y obtiene un array conteniendo los campos leídos. El delimitador de campo es una coma, a menos que se especifique otro delimitador con el tercer parámetro opcional.

fgetcsv (identificador_fichero, longitu, delimitador);



Una linea en blanco...

Una línea en blanco en un fichero CSV se devuelve como un array que contiene un único campo nulo, y esto no será tratado como un error.



¿No sabes lo que es CSV?

¿No sabes que es el formato CSV?
Pues el formato CSV es una serie de información que se encuentra delimitada por algún carácter en especial. Lo veremos mejor con un ejemplo, fíjate en la siguiente línea:

1,tcp,tcpmux,TCP Port Service Multiplexer [rfc-1078]

Esta línea sería un registro dividido en 4 campos, estando los campos divididos por comas. El delimitador en este caso son comas, pero podría ser cualquier cosa, incluso espacios en blanco. Lo importante es utilizar un identificador único, un identificador que no se encuentre en el contenido de cada uno de los campos.

Mismo ejemplo utilizando como delimitador el símbolo #

1#tcp#tcpmux#TCP Port Service Multiplexer [rfc-1078]

El siguiente programa explica como funciona la función fgetcscv(), pero no solo eso, ¿sabias que con fopen() podemos conectarnos a un URL y leer su contenido como si de un fichero se tratara?, increíble pero cierto.

PHP permite lo siguiente:

fopen("http://www.neohapsis.com/neolabs/neo-ports/neo-ports.csv","r"), es decir, podemos leer el contenido de cualquier URL, en este caso vamos a leer el contenido de un fichero CSV que se encuentra en el servidor www.neohapsis.com, ¡¡ no hace falta que el fichero se encuentre en nuestro disco duro !!. Este fichero remoto contiene la información de puertos en CSV (separados por comas). La función a utilizar es fgetcsv ($fp,1000,",");

El programa quedaría como sigue:

```
<?
$fichero="http://www.neohapsis.com/neolabs/neo-ports/neo-ports.csv";
$fp=fopen($fichero,"r");
while ($data = fgetcsv ($fp,1000,",")) {
print "Puerto:".$data[0]."     Tipo:".$data[1].
        "     Descripcion:".$data[3]."<br>";
}
fclose ($fp);
?>
```

```
Archivo  Edición  Ver  Favoritos  Herramientas  Ayuda
Atrás  Búsqueda  Favoritos  Multimedia
Dirección  Ir  Vínculos

Puerto:1   Tipo:tcp   Descripcion:TCP Port Service Multiplexer [rfc-1078]
Puerto:1   Tipo:udp   Descripcion:[trojan] Sockets des Troie
Puerto:1   Tipo:udp   Descripcion:TCP Port Service Multiplexer
Puerto:2   Tipo:tcp   Descripcion:Management Utility
Puerto:2   Tipo:tcp   Descripcion:[trojan] Death
Puerto:2   Tipo:udp   Descripcion:Management Utility
Puerto:3   Tipo:tcp   Descripcion:Compression Process
Puerto:3   Tipo:udp   Descripcion:Compression Process
Puerto:5   Tipo:tcp   Descripcion:Remote Job Entry
Puerto:5   Tipo:udp   Descripcion:Remote Job Entry
Puerto:7   Tipo:tcp   Descripcion:Echo
Puerto:7   Tipo:udp   Descripcion:Echo
Puerto:9   Tipo:tcp   Descripcion:Discard
Puerto:9   Tipo:udp   Descripcion:Discard
Puerto:11   Tipo:tcp   Descripcion:Active Users
Puerto:11   Tipo:udp   Descripcion:Active Users
Puerto:13   Tipo:tcp   Descripcion:Daytime
Puerto:13   Tipo:udp   Descripcion:Daytime
Puerto:15   Tipo:tcp   Descripcion:Netstat
Puerto:15   Tipo:tcp   Descripcion:[trojan] B2
Puerto:17   Tipo:tcp   Descripcion:Quote of the Day
Puerto:17   Tipo:udp   Descripcion:Quote of the Day
Puerto:18   Tipo:tcp   Descripcion:Message Send Protocol
Puerto:18   Tipo:udp   Descripcion:Message Send Protocol
Puerto:19   Tipo:tcp   Descripcion:Character Generator
Puerto:19   Tipo:udp   Descripcion:Character Generator
```

Comprobar si un fichero existe

Se puede comprobar si un fichero existe utilizando la función file_exists().

```
<?
$fichero="c:\\windows\\iis6.log";   //fichero en formato texto que se va a abrir.
if (file_exists($fichero)) {        // abre el fichero y crea un identificador.
Print ("El fichero existe");
}else {
print ("El fichero no existe");
}
?>
```

Es recomendable siempre que se trabaje con ficheros comprobar la existencia de estos para evitar errores.

Escritura en un fichero

Escribir en un fichero es la tarea más común cuando se utilizan los ficheros. Básicamente PHP ofrece dos funciones para añadir una cadena de texto a un fichero.

Ambas funciones tienen la misma funcionalidad, es decir, que son lo mismo.

fputs (identificador,cadena a escribir).
Fwrite (identificador, cadena a escribir).

Como las demás funciones el primer parámetro es el identificador del fichero abierto al que se le va a añadir la cadena de texto, el segundo parámetro es la cadena de texto a añadir.

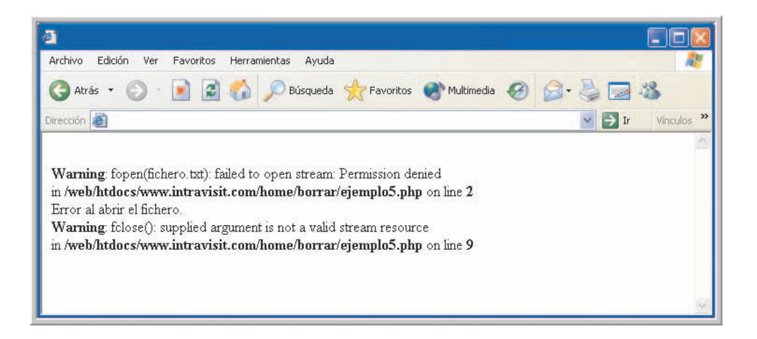
Como sucede con el resto de funciones si la operación se ha realizado con éxito devolverá TRUE en caso de error devolverá FALSE.

Para abrir un fichero como escritura es necesario comunicar el modo "a", en caso de que se intente añadir texto a un fichero que ha sido abierto en modo de lectura el intérprete de PHP devolverá un mensaje de error.

```
<? $fichero="fichero.txt";
if ($fd=fopen($fichero,"a")) {
for ($cuenta==1;$cuenta<=10;$cuenta=$cuenta+1) {
fwrite($fd,"Esta es la linea número: $contador\r\n");
}
} else {
print ("Error al abrir el fichero.");
}
fclose($fd);
?>
```

Este programa crea un fichero si no existe y escribe 10 lineas de texto.

Atención: Puede ocurrir que al crear o escribir en un fichero te de error por no tener permiso de escritura, la siguiente imagen es un ejemplo de ello.



Para evitar este error de permisos hay que dar permisos de escritura al directorio, si estás utilizando Windows no tendrás problemas pero en caso de Linux seguramente tendrás que dar permisos.

Si tienes todos los permisos correctamente tendrás un nuevo archivo al ejecutar el script, el contenido del fichero es:

Esta es la linea numero: 1
Esta es la linea numero: 2
Esta es la linea numero: 3
Esta es la linea numero: 4
Esta es la linea numero: 5
Esta es la linea numero: 6
Esta es la linea numero: 7
Esta es la linea numero: 8
Esta es la linea numero: 9
Esta es la linea numero: 10

Es un ejemplo muy sencillo pero suficiente para ver el funcionamiento. ¿Y ahora?, ¿te acuerdas del script del número anterior? El script que estudiamos en el número 15 de PC PASO A PASO generaba un listado de IPs y se mostraba en pantalla. Ahora vamos a modificar el script para que genere las IPs y se guarden en un fichero para que en posteriores script podamos acceder a las ips sin necesidad de volverla a generar.

```
<?php
$ipinicio=ip2long("81.35.0.0");
$ipfinal=ip2long("81.39.255.255");
for ($ip=$ipinicio;$ip<=$ipfinal;$ip=$ip+1) {
            if ($fd=fopen("ips.txt","a")) {
                        fwrite($fd,long2ip($ip)."\r\n");
            }
}
?>
```

De esta forma se ha creado un fichero llamado ps.txt y que contiene todas las IPS, un fichero con un tamaño de 4 megas (menudo fichero de ips).

Recorriendo un fichero

Hasta ahora las funciones que hemos visto permiten leer el contenido en su totalidad o una porción indicada. Puede resultar últil moverse por el fichero para leer determinados puntos o incluso para añadir texto en algunos lugares del fichero.

Para recorrer el fichero PHP ofrece nuevas funciones, rewind(), coloca la posición del puntero de acceso al fichero en la primera posición.

```
<?
$fichero="c:\\windows\\iis6.log";   // fichero en formato texto que se va a abrir.
if ($fp=fopen($fichero,"r")) {       // abre el fichero y crea un identificador.
Rewind($fp)                          // Nos situa en la primera posición del fichero.
} else {
print ("No se ha podido abrir el fichero");
}
?>
```

Para situarnos en una posición especifica se utiliza la posición fseek(). La función requiere de dos argumentos: como siempre el identificador del fichero y el número de caracteres que se desea saltar a partir del cual se comenzará el proceso de lectura o escritura.

```
<?
$fichero="c:\\windows\\iis6.log";   //fichero en formato texto que se va a abrir.
if ($fp=fopen($fichero,"w")) {        // abre el fichero – modo escritura.
fseek($fp,30);                        // Nos situa en el carácter número 30 del fichero.
fread($fp,40);                        // lee 40 caracteres desde la posición número 30.
} else {
print ("No se ha podido abrir el fichero");
}
?>
```

Para saber en que posición se encuentra el puntero se utiliza la función ftell(). Si se utiliza la función fseek() puede dar error si colocamos el puntero en un lugar superior al tamaño del fichero, para evitar esto se utiliza la función feof() que nos dirá si hemos llegado al final del fichero.

```
<?
$fichero="c:\\windows\\iis6.log";   //fichero en formato texto que se va a abrir.
 if ($fp=fopen($fichero,"r")) {       // abre el fichero – modo lectura.
While (feof($fp)) {
print (fgetc($fp);
} else {
print ("No se ha podido abrir el fichero");
}
?>
```

El ejemplo anterior abre el fichero iis6.log y lo lee carácter a carácter hasta que se llega al final indicado por feof().

Copiar, borrar y renombrar ficheros

Para copiar un fichero se utiliza la función copy(), se necesitan dos argumentos, el primero es el nombre del fichero a copiar y el segundo argumento es el nombre que se desea colocar a la copia del fichero.

```
<?
$fichero="c:\\windows\\iis6.log";
copy ($fichero,"c:\\iis6_copia.log");
?>
```

El ejemplo anterior copia un fichero llamado iis6.log en otro directorio y con el nombre iis6_copia.log, recuerda que tienes que tener permisos para copiar ficheros o te ocurrirá lo mismo que al crearlos.

Para eliminar un fichero, se utiliza la función unlink().

```
<?
$fichero="c:\\iis6_copia.log";
if (unlink($fichero)) {
print ("El fichero se ha eliminado con éxito");
} else {
print ("El fichero no se ha podido borrar.");
}
?>
```

Para renombrar un fichero se utiliza la función rename() siendo los dos argumentos necesarios, el nombre antiguo del fichero y el nuevo nombre.

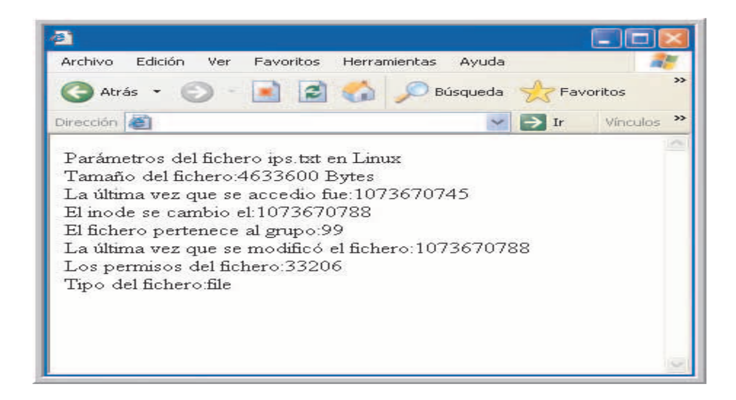
```
<?
$fichero_antiguo="c:\\iis6_copia.log";
$fichero_nuevo=="c:\\iis6_cambiado.log";
rename($fichero_antiguo,$fichero_nuevo);
?>
```

Atributos de un fichero

PHP proporciona funciones para obtener información acerca del propio fichero como el tamaño, tipo, ...

filesize() - proporcina el tamaño en bytes de un fichero.
fileatime() - Obtiene la última fecha de acceso a un fichero.
filectime() - Obtiene la fecha de cambio del inode del fichero.
filegroup() - Obtiene el grupo de un fichero.
filemtime() - Obtiene la fecha de modificación del fichero.
fileperms() - Obtiene los permisos del fichero.
filetype() - Obtiene el tipo de fichero.

```
<?php
$fichero="ips.txt";
print "Parámetros del fichero $fichero en Linux<br>";
print "Tamaño del fichero:".filesize($fichero)." Bytes<br>";
print "La última vez que se accedio fue:". fileatime($fichero)."<br>";
print "El inode se cambio el:".filectime($fichero)."<br>";
print "El fichero pertenece al grupo:".filegroup($fichero)."<br>";
print "La última vez que se modificó el fichero:".filemtime($fichero)."<br>";
print "Los permisos del fichero:".fileperms($fichero)."<br>";
print "Tipo del fichero:".filetype($fichero)."<br>";
?>
```



Por ahora ya hemos terminado con los ficheros, practica con ellos, recuerda que experimentar y ver lo que ocurres es la mejor forma de aprender, ¿con lo que has aprendido de PHP hasta ahora serías capaz de programar un sencillo escaneador de puertos?, pues seguro que sabes, una pista:

Para saber si una IP tiene el servicio Web bastaría con leer un fichero de la siguiente forma:
$fp=fopen(“217.174.193.62:80”,"r"), ¿ves por donde van los tiros?, venga, inténtalo y publica tu escaneador de puertos en el foro de la revista (www.hackxcrack.com).

David C.M

PROGRAMACION EN GNU LINUX
PROGRAMACION DEL SISTEMA: EL SISTEMA I.P.C.

el_chaman. LUIS U. RODRIGUEZ PANIAGUA

- Damos por terminado el curso de C y damos la bienvenida a la PROGRAMACIÓN DEL SISTEMA.
- I.P.C. (Comunicación entre Procesos)
- Identificando los procesos y sus dependencias
- Creando y Matando procesos

1. Presentación

Este es el primer artículo de programación de sistema que vamos a ver. El curso básico de C ha terminado oficialmente, pero no vamos a dejarle por el camino.

Tal vez de la impresión de que el curso de C ha sido rápido y quedan muchas cosas por explicar. Ante esto he de aclarar algunos puntos:

- Primero: Existe un artículo que no se publicará en la revista que analiza pormenorizadamente el código de la matriz vista en el anterior número, así como otras posibles soluciones. Será "el punto y seguido" al curso de C.

  Ese artículo está disponible para su descarga en http://leonov.servebeer.com:2525/~luis/descargas.php

  Además, en dicha página o en el foro de la revista atenderé gustosamente todas las dudas relativas al curso de programación en C.



Para cuando leas...

Para cuando leas estas letras, la nueva página Web de la revista estará 100% funcional. Podrás acceder mediante las direcciones www.hackxcrack.com y www.pcpasoapaso.com. Desde allí también podrás descargarte el artículo anteriormente citado y muchos otros ya liberados.

La razón principal de separar este último artículo de la serie de artículos que venían tratando el lenguaje C es la demanda que muchos lectores han hecho de temas que se centren más en la programación del sistema GNU/LiNUX que en el lenguaje C.

Esto **no** quiere decir que:

--> lo que vayamos a ver a partir de ahora sea más difícil. No lo es y trataré de plasmarlo lo más sencillo posible tal y como he hecho hasta la fecha.

--> dejemos de lado el lenguaje C. Si se ha dado el lenguaje C ha sido por una razón ya citada en anteriores artículos: Va a ser la principal herramienta que usemos para realizar nuestra programación del sistema. Necesitábamos una base sobre esta herramienta y ya la tenemos. Es hora de comenzar a utilizarla, tarea que cada día nos acostumbrará más a dicha herramienta: No vamos a dejar de ver el C y las cosas nuevas que puedan ir surgiendo se comentarán y resaltarán.

- Segundo: Un curso de lenguaje programación no sirve para nada si no se

programa. Esto que en principio parece una perogrullada es muy importante: Podemos tener cien cursillos de C bajados de internet y cincuenta libros. Si no hacemos programas, no nos servirán de nada. He procurado ir explicando conceptos con ejemplos. Una vez indicado el camino, sóis vosotros quién debéis recorrerlo. Muchos estamos dispuestos a acompañaros en este camino.

- Tercero: Insistiendo una vez más, os recuerdo que el que empecemos a dar ya programación del sistema **no quiere decir que vaya a ser más difícil que lo visto hasta ahora**. Sí anticipo que aparecerán nuevos conceptos importantísimos a la hora de programar el sistema y que trataré de exponerlos lo mejor que pueda.

Y creo que esto es todo. Esta es la presentación de artículo más larga que he hecho, pero había puntos como el de el artículo anexo a al código de la matriz y la terminación un tanto abrupta del curso del lenguaje C que me obligaban a dar una explicación y no abandonar a aquellos que han comenzado a aprender este lenguaje con esta serie de artículos. Espero que esta solución satisfaga tanto a los que quieren profundizar un poco más en el lenguaje C antes de meterse en harina, como a aquellos a los que el curso de C ya se les hacía pesado por conocido y desean ya darle a la amasadora.

A todos vosotros, un saludo, y muchísimas gracias por vuestras críticas y sugerencias.

Este artículo está dedicado a cierto pequeñín que se le ocurrió nacer en el 2003 y a quienes le están cuidando, cuya valentía debería admirarnos a todos.

2. Programación del sistema: El subconjunto I.P.C.

2.1. I.P.C. ¿Qué es eso?

I.P.C. es un acrónimo inglés que corresponde al término Inter Process Comunication (Comunicación Entre Procesos).

El I.P.C. es **tan sólo una parte** de la programación del sistema, concretamente la que trata con procesos y la forma en que estos se comunican entre sí.

Antes de comenzar a definir algunos conceptos importantes como proceso, voy a explicar brevemente porqué he escogido comenzar por esta parte de la programación del sistema.

La razón por la que empezaremos la programación del sistema por el subconjunto I.P.C. en vez de manejo de ficheros y directorios (ojo que hablamos del manejo de ficheros y directorios desde el punto de vista de la programación del sistema, no de lo visto en los primeros números del curso de Linux aunque sí relacionado), o el subconjunto de llamadas al **kernel**, es que es una parte a la vez bastante atractiva y a la vez bastante peliaguda. Esa parte nos permitirá realizar programas que exploten las principales características de nuestro S.O.: Multitarea real y comunicaciones en red

Dicho esto, pasemos a aclarar conceptos importantísimos para el tema que nos ocupa:

2.1.1. Procesos

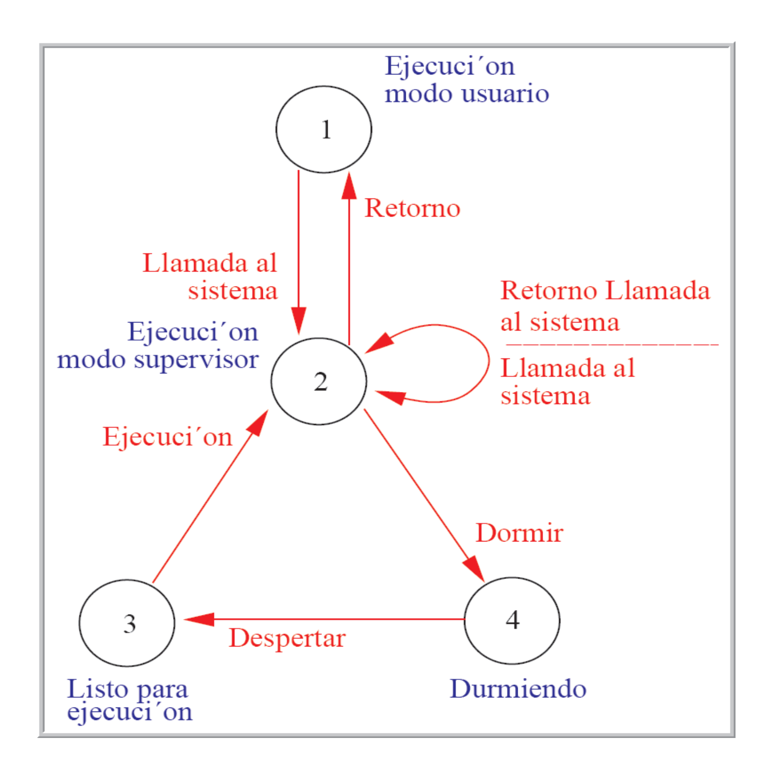
Hasta ahora hemos visto en los anteriores artículos como hacer nuestros propios programas. Estos programas los hemos hecho utilizando un lenguaje determinado, compilándolos y posteriormente ejecutándolos.

"Ejecutar un programa" es un término con el que estamos muy familiarizados, pero obviado en muchas ocasiones. ¿Qué ocurre cuando ejecutamos un programa? Lo que ocurre es que el núcleo carga el archivo ejecutable del disco en memoria y le añade cierta información adicional. En ese momento un programa se ha convertido en un proceso.

La información adicional que se le añade a un proceso es para que este pueda ser manejado por el núcleo.

Tras lo leído, es fácil deducir que un único ejecutable puede ser llamado varias veces, creado así varios procesos (copias en memoria) que hacen lo mismo pero que son independientes. Será el núcleo el encargado de gestionar como y cuando a un proceso le toca realizar sus funciones.

El **tiempo de vida de un proceso** es el conjunto de estados por los que un proceso puede pasar desde que comienza hasta que termina. En un entorno UNiX, los estados por los que puede pasar un proceso son:



Más adelante veremos y explicaremos en mayor profundidad la información asociada a este gráfico. Sólo quiero llamar la atención sobre lo que ocurre en el paso 2. Cuando hacemos una llamada al sistema, pasaremos a modo supervisor para poder realizar operaciones sobre el núcleo en modo supervisor. Este será el principal hueco que aprovecharemos a la hora de comprometer la seguridad de un sistema, obteniendo permisos bajo ciertas condiciones que a priori no tiene un usuario "normal".

Pero antes de llegar ahí, nos queda aún mucho camino por recorrer. Por ahora quedémonos con la idea que refleja dicho gráfico: Un proceso no es algo estático, sino que mientras este permanece en memoria ejecutándose, pasa por diversos estados. La razón de ello es que un proceso no está solo en memoria, sino que puede haber simultáneamente muchos procesos, todos compitiendo por obtener los mismos recursos del sistema (cpu, memoria, disco, I/O...). El núcleo será el encargado de gestionar qué procesos deben de hace qué y cuando. Es lo que se conoce como concurrencia y, como veremos en esta serie, también se nos suministrarán herramientas (semáforos) que nos permitirán gestionar la concurrencia de nuestros propios programas si así lo deseamos o necesitamos.

Una manera sencilla de ver los procesos que se están ejecutando en nuestro sistema es mediante el comando **ps**. Como este comando suele variar ligeramente en sus parámetros en distintos sabores UNiX, recomiendo la consulta de la página del manual (por ejemplo, en GNU/LiNUX, el comando **ps –A** lista todos los procesos del sistema. En *BSD, la orden es **ps -a** y aún así esto no garantiza que se muestren todos los procesos, dado que el sistema debe de estar configurado para ello.

Sea como fuere, una salida clásica en GNU/LiNUX suele ser:

```
luis@leonov:~$ ps -A
  PID TTY      TIME     CMD
    1 ?    00:00:05  init
    2 ?    00:00:00  keventd
    0 ?    00:00:00  ksoftirqd_CPU0
    0 ?    00:00:00  kswapd
    0 ?    00:00:00  bdflush
    0 ?    00:00:00  kupdated
   10 ?    00:00:00  khubd
   11 ?    00:00:11  kjournald
  121 ?    00:00:00  kjournald
  122 ?    00:00:00  kjournald
```

```
123 ?    00:00:00 kjournald
176 ?    00:00:00 eth0
196 ?    00:00:00 dhclient
200 ?    00:00:00 eth1
213 ?    00:00:00 portmap
503 ?    00:00:05 syslogd
506 ?    00:00:00 klogd
..........
```

Para obtener aún más información sobre estos procesos, añadimos el parámetro -f:

```
luis@leonov:~$ ps -Af // En *BSD ps -al

UID  PID  PPID  C  STIME TTY      TIME    CMD
root   1    0   0  16:42  ?     00:00:05  init [2]
root   2    1   0  16:42  ?     00:00:00  [keventd]
root   0    1   0  16:42  ?     00:00:00  [ksoftirqd_CPU0]
root   0    1   0  16:42  ?     00:00:00  [kswapd]
root   0    1   0  16:42  ?     00:00:00  [bdflush]
root   0    1   0  16:42  ?     00:00:00  [kupdated]
root  10    1   0  16:42  ?     00:00:00  [khubd]
root  11    1   0  16:42  ?     00:00:11  [kjournald]
root 121    1   0  16:43  ?     00:00:00  [kjournald]
root 122    1   0  16:43  ?     00:00:00  [kjournald]
root 123    1   0  16:43  ?     00:00:00  [kjournald]
root 176    1   0  16:43  ?     00:00:00  [eth0]
......
```

La información relevante mostrada será:

UID

(User Identification) Nos muestra qué usuario lanzó ese proceso.

PID

(Process Identification) Nos muestra el número identificador del proceso. Este número es único para cada proceso y actúa de DNI de cada uno. De esta manera el sistema podrá manejar los procesos utilizando su DNI.

PPID

(Parent Process Identification) Identificador del proceso padre. En un sistema UNiX los procesos son creados siempre por otros procesos. Esto hace que siempre exista una relación padre-hijo entre ellos y que obtengamos una estructura arbórea de dependencias entre procesos.



Existe un programa llamado **pstree** que muestra esta estructura arbórea de dependencias padre-hijo. Suele estar disponible en todos los UNiX, pero no se suele instalar por defecto. Hasta donde se, en Debian se puede instalar mediante **apt-get install pstree** y en *BSD se dispone de un port en **/usr/ports/sysutils/pstree/**. Sea como fuere, lo podemos descargar de ftp://ftp.thp.Uni-Duisburg.DE/pub/source/

He dicho siempre. ¿Seguro? Siempre existirá un único proceso padre de todos los demás. a la vista del comando **ps** y **pstree**, podemos decir que este proceso "Adán" es **init**.

El resto de los campos de información los ignoraremos por ahora. Con lo visto, debe de quedar claro que la información más importante asociada a un proceso será su **PID** y su **PPID**. Estos número que como hemos dicho identifican respectivamente al proceso, y al proceso padre del mismo, son números de 16 bits que serán asignados secuencialmente por el núcleo cuando se crea un proceso.

Cuando programamos, el tipo asociado a un **PID** suele ser un entero **(int)**. Dado que siempre debemos de programar de manera que nuestro código sea lo más portable posible, se suele emplear el tipo **pid_t** proporcionado por **<sys/types.h>**.

Veamos como podemos hacer que nuestros programas conozcan su **PID** y su **PPID**:

```
////////////////////////////////////////////////////////////////////////////////////////
//
// ARCHIVO: pid.c
//
// DESCRIPCIÓN:
//   Este archivo muestra como obtener el PID y el PPID de
//   un programa.
//
// COMPILACIÓN:
//   gcc -o pid pid.c
//
////////////////////////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main()
{
        pid_t mi_pid, pid_papa;
        mi_pid = getpid();
        pid_papa = getppid();
        printf("\n Mi PID es %d y el de mi padre %d\n", mi_pid, pid_papa);
}
```

Un experimento interesante será el añadir al final del programa mostrado una instrucción como **sleep(10)**;. Esta instrucción hace que tras ejecutar nuestro programa, este espere 10m segundos para terminar. Si en un terminal ejecutamos este programa, y en otro ejecutamos **ps** con los parámetros adecuados, podremos ver algo similar a:

```
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./pid2
Mi PID es 24560 y el de mi padre 24490
```

Y en la otra ventana:

```
luis@leonov:~/hxc/articulo9_el_chaman$ ps -Al
F S   UID   PID    PPID  C PRI  NI ADDR SZ  WCHAN TTY        TIME    CMD
...........
 0 S  1004 24560 24490  0  69     0     -  334   126840   pts/3       00:00:00 pid2
...........
```

Los datos pueden variar un poco, pero lo importante es observar que el **PID** y el **PPID** coinciden.

Ya sabemos qué es un proceso y sabemos cómo se identifica. A continuación vamos a pasar al lado siniestro de los entornos UNiX-like: El comando **kill**. Este comando sirve para matar cualquier proceso que haya creado nuestro usuario (el usuario **root** puede matar cualquier proceso del sistema). Este comando funciona enviando la señal **SIGTERM** a nuestro proceso, haciendo que este muera de asesinato alevoso.

Sobre señales hablaremos más adelante, pero anticipemos que es el medio de comunicación más básico que existe entre procesos. Todos los procesos entienden un "lenguaje" especial llamado señales. Mediante estas señales que son órdenes de este lenguaje, podremos dar ciertas órdenes a los procesos. Así que estrictamente no estamos asesinando un proceso: Le estamos invitando cortésmente a que se suicide.

Vamos a ver esto con un ejemplo:

```
////////////////////////////////////////////////////////////////////
//
// ARCHIVO: pesado.c
//
// DESCRIPCIN:
//   Este archivo nos hará perder la paciencia
//
// COMPILACIN:
//   gcc -o pesado pesado.c
//
////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
```

```
int main()
{
 while(1)
 {
  printf(" \a %d dice:  Desisme, ¿qué le pasó al mamut?\n", (pid_t)getpid());
  sleep(1);
 }
}
```

Este programa es exactamente lo que parece: Un bucle infinito. Este programa repite mientras cierto, un absurdo mensaje acompañado por un pitido en el altavoz (secuencia de escape **\a**). Para el que no crea que este programa no es digno merecedor de su nombre, ejecutadlo durante diez minutos. Si pasado un tiempo no os ha dado un ataque de tics o unas "ligeras" ganas de dejar mudo al ordenador para siempre, entonces es que tenéis mucha paciencia.

Una posible salida de este programa será:

```
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./pesado
  27972 dice:  Desisme, ¿qué le pasó al mamut?
  27972 dice:  Desisme, ¿qué le pasó al mamut?
  27972 dice:  Desisme, ¿qué le pasó al mamut?
........
```

El número que aparecerá a la izquierda (en el ejemplo el 27972), es el **PID** del programa.

Ahora, desde otro terminal, ejecutemos

```
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ kill 27972
```

En el terminal que nos estaba molestando aparecerá:

```
........
  27972 dice:  Desisme, ¿qué le pasó al mamut?
  27972 dice:  Desisme, ¿qué le pasó al mamut?
Terminado
```

Descanse en paz.

En ocasiones, los procesos se aferran a la vida. Son así, qué le vamos a hacer. Entonces conviene invocar al comando **kill** con el parámetro **-9**, lo que provocará la ejecución de un **kill** que no podrá ser ignorado. Es buena costumbre utilizar siempre este parámetro. Como curiosidad, decir que la salida ahora será:

```
........
  27972 dice:  Desisme, ¿qué le pasó al mamut?
  27972 dice:  Desisme, ¿qué le pasó al mamut?
Terminado (killed)
```

Lo que quiere decir que, esta vez sí, ha sido un asesinato y no un suicidio.

Tras leer tanto humor (malo) negro, puede parecer que esto de suicidarse y matar es un recurso para hacer entretenido el artículo. Nada más lejos de la realidad: No es lo mismo que un proceso se mate a él mismo aunque sea por orden de otros, a que otros lo maten. Es una sutil diferencia que nos debe de hacer sospechar que aunque el efecto sea el mismo, lo que ha ocurrido por debajo no lo es tanto. Ya lo veremos más adelante.



En varias ocasiones he visto preguntar a la gente si no hay alguna manera para matar los procesos por el nombre del programa en vez de por el **PID**. A priori, con los comandos vistos hasta ahora, no. Pero existe un comando llamado **skill** que sí puede hacer esto. Este comando tampoco se suele instalar por defecto pero está disponible en todas las distribuciones que conozco. Si por alguna razón no lo estuviese, podemos bajarnos las fuentes de ftp://fast.cs.utah.edu/pub/skill/.

Claro que siempre podemos hacernos nuestra propia versión de **skill**.... Esto no es una machada, y es un ejercicio muy interesante. Para ello disponemos del directorio **/proc** donde está la información disponible que nos muestra el comando **ps** (¿Recordáis lo que decíamos

en los primeros capítulos de que en UNiX TODO es un archivo? Vamos a verlo.).

Ejecutemos en un terminal de nuevo el programa **pesado**. Ahora tendrá otro **PID**:

```
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./pesado
  28444 dice:  Desisme, ¿qué le pasó al mamut?
........
```

Ahora nos vamos al directorio **/proc/28444/**, que es el lugar donde encontraremos toda la información relativa al proceso **28444**. Si curioseamos un poco por los archivos que nos encontramos, veremos que se nos da información sobre los dispositivos montados, la carga de CPU, etc... Es decir, toda la información que le puede resultar de interés a un proceso. Vemos, además, que existe un archivo llamado **cmdline**, que contiene el nombre del programa correspondiente a dicho proceso. Juntando esta información, veremos que es relativamente sencillo hacer un programa que obtenga el **PID** (nombre del directorio) de un determinado programa (lo que se encuentre en el archivo **cmdline**) de dicho directorio. Sería una tarea de ir recorriendo todos los directorios, almacenar temporalmente el nombre (número) de directorio y ver si lo contenido en el archivo **cmdline** de dicho directorio coincide con lo que buscábamos. Si es así ya tenemos el número de proceso y lo podremos matar. Si no, podemos seguir buscando.

El código de **skill** podría ayudarnos a realizar esta tarea, pero si lo echamos un vistazo tal vez nos asuste un poco y no podamos identificar de un primer vistazo estas tareas. Aún así es una tarea que recomiendo, sobre todo cuando veamos las señales.

Pero antes de ver señales, para poder trabajar más a gusto con ellas, vamos a ver como podemos crear nuevos procesos desde nuestros propios programas. Bienvenidos al apasionante mundo de la clonación.

2.1.2. Creación de procesos

A la hora de generar nuestros propios procesos, tendremos varias posibilidades:

system

La primera manera de crear nuevos procesos es invocar a la función system perteneciente a la librería estándar de C. Con esta función podremos invocar cualquier comando o programa del sistema que se encuentre en el **path**. Esta función, en caso de error, retorna **-1**. Si no puede ejecutar el programa por falta de permisos, devuelve 127. Un ejemplo:

```
////////////////////////////////////////////////////////////
//
// ARCHIVO: system.c
//
// DESCRIPCIóN:
//   Ejemplo de uso de system
//
// COMPILACIÓN:
//   gcc -o system system.c
//
////////////////////////////////////////////////////////////

#include <stdio.h>
int main()
{
        int retorno;
        retorno = system("ls -l");
        return retorno;
}
```

exec

La familia de funciones **exec** (llamada así porque no existe una única función **exec**) nos permitirá ejecutar otros programas desde nuestro propio código. Cuando realizamos una llamada **exec**, nuestro programa para su ejecución y comienza la ejecución del programa que llamamos mediante **exec**. El programa original **deja de ser accesible**, no pudiendo retornar a él.

La familia de funciones **exec** puede ser clasificada como:

1.- Funciones que contienen la letra **p** en sus nombres (**execvp** y **execlp**). Sus declaraciones son:

- int execvp(char *file, char *argv[]);
- int execlp(char *file, char *arg0, char *arg1, ..., char*argN, (char *) NULL);

Este tipo de funciones busca el programa a ejecutar pasado a ***file** en el **path**. Como veremos ahora, las funciones que no llevan la letra **p** en el nombre, requieren que se de la ruta (path) absoluta al programa a ejecutar.

***arg0, *arg1, ..., *argN** son punteros a cadenas de texto que formarán los argumentos a pasar al programa. Se suele apuntar ***arg0** al nombre del propio programa (donde apunta ***file**), y se requiere que el último puntero apunte a **NULL**.

***argv[]** será otra forma de representar los argumentos pasados al programa. Es un vector de cadenas de texto que contiene los parámetros. Por convenio, **argv[0]** apunta siempre al nombre del propio programa y **argv[N]** a **NULL**

2.- Funciones que contienen la letra **v** en sus nombres (La ya vista **execvp** más **execv** y **execve**. Sus declaraciones son:

- int execv(char *path, char *argv[]);
- int execve(char *path, char *argv[], char *envp[]);

Estas funciones se caracterizan porque los argumentos del programa se pasan mediante ***argv[]** arriba explicado.

En cuanto al nuevo parámetro que aparece, ***envp[]**, representa una lista de variables de entorno que podemos pasar al nuevo programa. Estas variables de entorno se pasan como cadenas de texto con el formato **VARIABLE=valor. env[N]** debe valer **NULL**.

3.- Funciones que contienen la letra **l** (**execl**, **execle** y la ya vista **execlp**. Su declaración es:

- int execl (char *path, char *arg0, char *arg1, ..., char *argN, (char *)NULL);
- int execle(char *path, char *arg0, char *arg1, ..., char *argN, (char *)NULL, *envp[]);

Este tipo de funciones pasa los argumentos del programa uno a uno, a diferencia del caso anterior que se han de meter previamente en una lista de argumentos.

4.- Funciones que contienen la letra **e** (**execve** y **execle**, ambaas vistas).

Este tipo de funciones poseen un argumento adicional, **char *envp[]** que contiene las variables de entorno a pasar al programa a ejecutar.

Tanta variedad principalmente es debida a cómo pasamos los argumentos al programa. Y esto es importante porque nuestros programas también pueden recibir parámetros de la línea de comandos. Como esto en el cursillo de C básico se obvió, vamos a explicarlo brevemente con unos ejemplos.

En ocasiones nos interesará pasar a nuestros programas diversos argumentos. Por ejemplo:

$ codifica archivo.txt

archivo.txt es un argumento de nuestro programa **codifica**. Puede ser el nombre de cualquier archivo. La manera de que "veamos" desde dentro del código, el nombre de cualquier archivo, es consultando la lista de argumentos. Esta está disponible en **argv[]** que es un vector de cadenas de texto que contendrá tantas cadenas como argumentos se pasen. Veamos un código que nos muestra sus argumentos:

```
////////////////////////////////////////////////////////////
//
// ARCHIVO: args.c
//
// DESCRIPCIÓN:
//  Este programa nos muestra el paso de argumentos
//  por línea de comandos
//
// COMPILACIÓN:
//  gcc -o args args.c
//
////////////////////////////////////////////////////////////
```

```
#include <stdio.h>
int main(int argc, char *argv[])
{
      // argc contiene el número de parámetros que
      // se pasan arg[0] es siempre el nombre del
      // programa. Imprimimos todos los parámetros
      //que se nos pasen
      int i;

       for(i=0;i < argc;i++)
            printf("Argumento %d : %s\n", i, argv[i]);
}
```

Este programa nos proporcionará esta salida:

```
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./args 1 2 3 4
Argumento 0 : ./args
Argumento 1 : 1
Argumento 2 : 2
Argumento 3 : 3
Argumento 4 : 4
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./args uno dos tres cuatro
Argumento 0 : ./args
Argumento 1 : uno
Argumento 2 : dos
Argumento 3 : tres
Argumento 4 : cuatro
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./args uno dos

Argumento 0 : ./args
Argumento 1 : uno
Argumento 2 : dos
```

Creo que con este ejemplo queda explicado como funcionan los parámetros por línea de comandos.

Dicho esto, se puede entender porqué puede ser muy útil el tener funciones **exec** que reciban **argv[]** como parámetro, sobre todo si queremos pasar al nuevo programa que ejecutemos los mismos parámetros que ha recibido el nuestro.

Veamos como podríamos hacernos nuestra propia versión del comando **ls**:

```
//////////////////////////////////////////////////////////////////////////////////
// ARCHIVO: mils.c
//
// DESCRIPCIÓN:
//   Otro programa que hace ls
//
// COMPILACIÓN:
//   gcc -o mils mils.c
//////////////////////////////////////////////////////////////////////////////////

#include <stdio.h>
int main(int argc, char *argv[])
{
      // Ejecutamos ls con los parámetros que se nos
      // pasan desde mils
      execvp("ls",argv);
}
```

Y ahora es cuando veremos el resultado:

```
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./mils
a.out  args  args.c  mils  mils.c  pesado  pesado.c  pid  pid2  pid2.c  pid.c
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./mils -l
total 92
-rwx------   1 luis   luis     11349 2003-12-19 18:05 a.out
-rwx------   1 luis   luis     11413 2003-12-19 17:56 args
-rw-------   1 luis   luis       522 2003-12-19 17:56 args.c
-rwx------   1 luis   luis     11349 2003-12-19 18:05 mils
-rw-------   1 luis   luis       331 2003-12-19 18:05 mils.c
-rwx------   1 luis   luis     11626 2003-12-19 15:43 pesado
-rw-------   1 luis   luis       389 2003-12-19 15:47 pesado.c
-rwx------   1 luis   luis     11609 2003-12-19 15:25 pid
-rwx------   1 luis   luis     11715 2003-12-19 15:25 pid2
-rw-r--r--   1 luis   luis       228 2003-12-19 15:24 pid2.c
-rw-r--r--   1 luis   luis       216 2003-12-19 15:24 pid.c
```

Hemos pasado los argumentos que se pasaban a nuestro programa al programa que queríamos ejecutar (**ls**), haciendo que éste se comporte como si le hubiésemos pasado los argumentos adecuados. ¿Por qué existen entonces funciones que hacen esto de otra manera?

Pues para que nosotros desde dentro del programa especifiquemos los parámetros:

```
////////////////////////////////////////////////////////////
// ARCHIVO: mils2.c
//
// DESCRIPCIÓN:
//   Otro programa que hace ls
//
// COMPILACIÓN:
//   gcc -o mils2 mils2.c
////////////////////////////////////////////////////////////

#include <stdio.h>
int main()
{
        // Ahora desde dentro del programa especificamos los
        // parámetros
        // Recordemos que arg0 tiene que llamarse igual que
        // el programa.
        execlp("ls", "ls", "-l", "-h", NULL);
}
```

Lo cual provocará la salida:

```
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./mils2
total 108K
-rwx------   1 luis   luis    12K 2003-12-19 18:05 a.out
-rwx------   1 luis   luis    12K 2003-12-19 17:56 args
-rw-------   1 luis   luis    522 2003-12-19 17:56 args.c
-rwx------   1 luis   luis    12K 2003-12-19 18:05 mils
-rwx------   1 luis   luis    12K 2003-12-19 18:16 mils2
-rw-------   1 luis   luis    331 2003-12-19 18:16 mils2.c
-rw-------   1 luis   luis    331 2003-12-19 18:05 mils.c
-rwx------   1 luis   luis    12K 2003-12-19 15:43 pesado
-rw-------   1 luis   luis    389 2003-12-19 15:47 pesado.c
-rwx------   1 luis   luis    12K 2003-12-19 15:25 pid
-rwx------   1 luis   luis    12K 2003-12-19 15:25 pid2
-rw-r--r--   1 luis   luis    228 2003-12-19 15:24 pid2.c
-rw-r--r--   1 luis   luis    216 2003-12-19 15:24 pid.c
```

Ahora podemos ir practicando con el resto de las funciones.

fork

La función **fork** creará un nuevo proceso en UNiX mediante la clonación del proceso que llama a dicha función. Es decir, si en un proceso invocamos a esta función, se crea un proceso **totalmente idéntico** llamado proceso hijo. Al proceso que llama originalmente a **fork** se le denomina proceso padre, y al proceso clonado, proceso hijo. **fork** retornará el **PID** del proceso hijo para el proceso padre y 0 para el proceso hijo.

Veamos un ejemplo:

```
////////////////////////////////////////////////////////////
// ARCHIVO: fork00.c
//
// DESCRIPCIóN:
//   Primer ejemplo de uso de fork
//
// COMPILACIÓN:
//   gcc -o fork00 fork00.c
////////////////////////////////////////////////////////////

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    int pid_hijo;
    int i;
    printf("Comienza el programa principal %d\n", (int) getpid());
    pid_hijo = fork();
    if (pid_hijo != 0) // Si somos el proceso padre (e.d. El pid_hijo
    {                  // vale distinto de 0
      for(i=0;i<3;i++)
            printf("Yo soy el padre. Mi PID es %7d\n", (int) getpid());
    }
    else
    {             // Somos el proceso hijo
      for(i=0;i<3;i++)
            printf("¡Yo soy Espartaco! (%7d)\n", (int) getpid());
    }
}
```

Una salida posible de este programa es:

```
Comienza el programa principal 8079
Yo soy el padre. Mi PID es    8079
Yo soy el padre. Mi PID es    8079
Yo soy el padre. Mi PID es    8079
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ¡Yo soy Espartaco! (   8080)
¡Yo soy Espartaco! (   8080)
¡Yo soy Espartaco! (   8080)
```

Como vemos cada proceso intenta ejecutar su parte cuando puede, sin orden alguno. (esto se ve más claramente si en el **for** sustituimos el **3** por una cifra mayor.

Lo deseable sería que pudiésemos controlar cuando y como se ejecuta cada uno de estos procesos. En posteriores artículos veremos que existe un mecanismo muy sofisticado denominado **semáforos** que nos permitirá hacer esto. Sin embargo, sin ser tan sofisticados, sí disponemos ahora dos mecanismos de sincronización entre procesos bastante sencillo: Las funciones **exit** y **wait**.

exit(int codigo) terminará la ejecución de un proceso devolviendo al sistema el valor especificado en código. Si se ejecuta en un programa hijo, pueden ocurrir dos cosas:

- Que el padre esté en estado **wait**. En este caso se notifica al proceso padre la terminación del proceso hijo pasándole los 8 bits menos significativos de **código**; de esta manera el proceso padre puede conocer en qué estado terminó el proceso hijo.
- Que el proceso padre no esté en estado **wait**, transformándose el proceso hijo en un proceso **zombie**. Este tipo de procesos es aquel que ocupa un lugar e la tabla de procesos de sistema y cuyo contexto ha sido descargado de memoria. Esto quiere decir, que aunque termine el proceso padre, el proceso hijo sigue estando muerto. El encargado de matar al proceso **zombie** será el proceso **init** del sistema.

```
//////////////////////////////////////////////////////////////////////
// ARCHIVO: zombi.c
//
// DESCRIPCIóN:
//   Creando zombis
//
// COMPILACIÓN:
//   gcc -o zombi zombi.c
//////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
        int pid_hijo;
        int i;
        pid_hijo = fork();
        if (pid_hijo != 0)
        {
                // Proceso padre sin wait()
                printf("Yo soy el padre. Mi PID es %7d\n", (int) getpid());
              // Simulamos un proceso de larga duración
              // haciendo dormir al proceso 1 minuto
                sleep(60);
        }
        else
        {
                // El proceso hijo imprime un mensaje y
                // termina inmediatamente mediante exit
                printf("Proceso hijo, candidato a Zombi (%7d)\n", (int) getpid());
                exit(0);
        }
}
```

Si mientras se está ejecutando ese proceso en otro terminal ejecutamos el comando ps observaremos lo siguiente:

```
F S  UID   PID  PPID  C PRI  NI ADDR SZ  WCHAN TTY      TIME    CMD
........
0 S  1004 11882 24490  0  69   0      -    334 126840   pts/3   00:00:00  zombi
1 Z  1004 11883 11882  0  69   0      -      0   1206cd  pts/3   00:00:00  zombi <defunct>
........
```

En la columna S (State) vemos que el proceso padre está durmiendo (S) y el proceso hijo, además de difunto, es un zombi (Z).

La otra posibilidad es que el proceso padre haya ejecutado la función **wait** pasando al estado **wait**. Esto quiere decir que la ejecución de un proceso se detiene hasta que alguno de sus procesos hijo termina. El funcionamiento de **wait** es el siguiente:

```
int pid_hijo, estado;
...
pid_hijo = wait(&estado);
...
```

Donde **pid_hijo** es el **PID** del proceso hijo zombi y estado es el estado en el que este realizó su salida.

El siguiente ejemplo muestra como modificaríamos el programa original:

```
//////////////////////////////////////////////////////////////////////////////////////////////
// ARCHIVO: fork01.c
//
// DESCRIPCIóN:
//   Primer ejemplo de sincronozación de procesos mediante
//   el uso de wait
//
// COMPILACIÓN:
//   gcc -o fork01 fork01.c
//////////////////////////////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
       int pid_hijo, estado;
       int i;
      printf("Comienza el programa principal %d\n", (int) getpid());
       pid_hijo = fork();
      if (pid_hijo != 0) // Si somos el proceso padre (e.d. El pid_hijo
       {                // vale distinto de 0
          for(i=0;i<3;i++)
              printf("Yo soy el padre. Mi PID es %7d\n", (int) getpid());
          pid_hijo=wait(&estado);
         printf("El proceso hijo %7d terminó con estado %d\n", pid_hijo, estado);
       }
       else
       {                // Somos el proceso hijo
          for(i=0;i<3;i++)
              printf("¡Yo soy Espartaco! (%7d)\n", (int) getpid());
          exit (0);
       }
}
```

Ahora la salida producida será:

```
luis@leonov:~/hxc/articulo9_el_chaman/codigo$ ./fork01
Comienza el programa principal 12291
Yo soy el padre. Mi PID es   12291
Yo soy el padre. Mi PID es   12291
Yo soy el padre. Mi PID es   12291
¡Yo soy Espartaco! (  12292)
¡Yo soy Espartaco! (  12292)
¡Yo soy Espartaco! (  12292)
El proceso hijo   12292 terminó con estado 0
```

Que es un comportamiento mucho más civilizado del que obtuvimos la vez pasada.

Al ver estos tres tipos de generación de procesos (ya sea mediante la ejecución de otros programas, ya sea mediante la creación real de proceso) tal vez los que vengáis de la programación en windows echéis de menos la función **spawn**. Esta función, que no existe en UNiX, funciona de manera similar a **exec** con la salvedad de que **exit**, como hemos visto, sustituye la ejecución del proceso original por el nuevo programa que mandamos ejecutar, sin posibilidad de retornar al proceso original.

spawn sí permite retornar al proceso original. Aunque la función **spawn** no exista, su funcionamiento, tras lo visto hoy será fácil de simular:

```
//////////////////////////////////////////////////////////////////////
// ARCHIVO: spawn.c
//
// DESCRIPCIóN:
//   Programa que emula la funcionaldad de la función spawn
//   perteneciente a la  API de win32
//
// COMPILACIÓN:
//   gcc -o spawn spawn.c
//////////////////////////////////////////////////////////////////////
```

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

/////////////////////////////////////////////////////////////////////
// Función: spawn
//
// Descripción:
//   Proporciona la misma funcinalidad que la
//   función homóloga
//   disponible en la API win32
/////////////////////////////////////////////////////////////////////

int spawn(char *programa, char *lista_argumentos[])
{
        pid_t pid_hijo;

        pid_hijo = fork();
        if(pid_hijo!=0)
        {
                // Proceso padre
                return pid_hijo;
        }
        else
        {
                // Proceso hijo
                // Ejecutamos el programa utilizando execvp
                // que ahora será un proceso independiente
                execvp(programa, lista_argumentos);

                // La función execvp sólo retorna en caso
                // de error, por lo que sólo se imprimirá
                // la siguiente línea si hay error en execvp
                printf("ERROR (spawn)");
                return -1;
        }
}
/////////////////////////////////////////////////////////////////////
// P r o g r a m a   p r i n c i p a l
/////////////////////////////////////////////////////////////////////

int main()
{
        char *listaargs[]={
                "ls",
                "-l",
                NULL
        };

        spawn("ls", listaargs);

        printf("El programa principal todavía sigue vivo\n");

        return 0;
}
```

3. Despedida

En el artículo de hoy nos hemos comenzado a familiarizar con los procesos UNiX. Conviene que vayamos practicando y experimentando con lo visto, dado que va a ser la base de todo lo que hagamos a partir de ahora. En los artículos trataremos propiamente los mecanismos de IPC que trabajan con los procesos. Haremos especial hincapié en las señales y los semáforos. También veremos cómo usar la memoria compartida o las colas de mensajes. Todos estos temas requieren de una explicación más detallada de cómo funcionan los procesos y es a lo que nos dedicaremos en el próximo artículo junto con la explicación de señales.

Tal vez alguien haya echado de menos los threads (hilos). Efectivamente he evitado el tema por la siguiente cuestión: Los hilos son un mecanismo que al igual que los procesos nos permitirán que un mismo programa realice varias tareas concurrentemente. La diferencia es que, a pesar de estar íntimamente ligados a los procesos (todos los procesos llevan al menos un hilo), su trato no es tan sencillo y requiere que tengamos cierta base en cómo funcionan los procesos. Por ello he decidido que será mejor ir sobre seguro y presentar posteriormente los hilos.

Bibliografía

Fco. Manuel Márquez, UNIX Programación Avanzada, 2ª Edición, Ra-Ma.

Mark Mitchell, Jeffrey Oldham, Alex Samuel, Advanced LiNUX Programming, First Edition, New Riders.

Antonio P. Giménez Lorenzo, UNIX System V, 1ª Edición , Anaya Multimedia.

Jorge E. Pérez Martínez, Programación Concurrente, 2ª Edición corregida, Editorial Rueda.

Juan M. Morera Pascual , Juan A. Pérez-Campanero Atanasio, Teoría y diseño de Sistemas Operativos, 1ª Edición, Anaya Multimedia.

Páginas del manual UNiX , Varios .

SERVIDOR DE HXC MODO DE EMPLEO

- Hack x Crack ha habilitado tres servidores para que puedas realizar las prácticas de hacking.

- Las IPs de los servidores de hacking las encontrarás en EL FORO de la revista (www.hackxcrack.com). Una vez en el foro entra en la zona COMUNICADOS DE HACK X CRACK (arriba del todo) y verás varios comunicados relacionados con los servidores. No ponemos las IP aquí porque es bueno acostumbrarte a entrar en el foro y leer los comunicados. Si hay alguna incidencia o cambio de IP o lo que sea, se comunicará en EL FORO.

- Actualmente tienen el BUG del Code / Decode. La forma de "explotar" este bug la explicamos extensamente en los números 2 y 3. Lo dejaremos así por un tiempo (bastante tiempo ;) Nuestra intención es ir habilitando servidores a medida que os enseñemos distintos tipos de Hack.

- En los Servidores corre el Windows 2000 con el IIS de Servidor Web. No hemos parcheado ningún bug, ni tan siquiera el RPC y por supuesto tampoco hemos instalado ningún Service Pack. Para quien piense que eso es un error (lógico si tenemos en cuenta que el RPC provoca una caída completa del sistema), solo decirte que AZIMUT ha configurado un firewall desde cero que evita el bug del RPC, (bloqueo de los puertos 135 (tcp/udp), 137 (udp), 138 (udp), 445 (tcp), 593 (tcp)). La intención de todo esto es, precisamente, que puedas practicar tanto con el CODE/DECODE como con cualquier otro "bug" que conozcas (y hay cientos!!!). Poco a poco iremos cambiando la configuración en función de la experiencia, la idea es tener los Servidores lo menos parcheados posibles pero mantenerlos operativos las 24 horas del día. Por todo ello y debido a posibles cambios de configuración, no olvides visitar el foro (Zona Comunicados) antes de "penetrar" en nuestros servidores.

- Cada Servidor tiene dos unidades (discos duros duros):
* La unidad c: --> Con 40GB y Raíz del Sistema
* La unidad d: --> Con 40GB
* La unidad e: --> CD-ROM

Nota: Raíz del Servidor, significa que el Windows Advanced Server está instalado en esa unidad (la unidad c:) y concretamente en el directorio por defecto \winnt\ Por lo tanto, la raíz del sistema está en c:\winnt\

- El IIS, Internet Information Server, es el Servidor de páginas Web y tiene su raíz en c:\inetpub (el directorio por defecto)

Nota: Para quien nunca ha tenido instalado el IIS, le será extraño tanto el nombre de esta carpeta (c:\inetpub) como su contenido. Pero bueno, un día de estos os enseñaremos a instalar vuestro propio Servidor Web (IIS) y detallaremos su funcionamiento.

De momento, lo único que hay que saber es que cuando TÚ pongas nuestra IP (la IP de uno de nuestros servidores) en tu navegador (el Internet explorer por ejemplo), lo que estás haciendo realmente es ir al directorio c:\Inetpub\wwwroot\ y leer un archivo llamado default.htm.

Nota: Como curiosidad, te diremos que APACHE es otro Servidor de páginas Web (seguro que has oído hablar de él). Si tuviésemos instalado el apache, cuando pusieses nuestra IP en TU navegador, accederías a un directorio raíz del Apache (donde se hubiese instalado) e intentarías leer una página llamada index.html ... pero... ¿qué te estoy contando?... si has seguido nuestra revista ya dominas de sobras el APACHE ;)

Explicamos esto porque la mayoría, seguro que piensa en un Servidor Web como en algo extraño que no saben ni donde está ni como se accede. Bueno, pues ya sabes dónde se encuentran la mayoría de IIS (en \Inetpub\) y cuál es la página por defecto (\Inetpub\wwwroot\default.htm). Y ahora, piensa un poco… … ¿Cuál es uno de los objetivos de un hacker que quiere decirle al mundo que ha hackeado una Web? Pues está claro, el objetivo es cambiar (o sustituir) el archivo default.html por uno propio donde diga "hola, soy DIOS y he hackeado esta Web" (eso si es un lamer ;)

A partir de ese momento, cualquiera que acceda a ese servidor, verá el default.htm modificado para vergüenza del "site" hackeado. Esto es muy genérico pero os dará una idea de cómo funciona esto de hackear Webs ;)

- Cuando accedas a nuestro servidor mediante el CODE / DECODE BUG, crea un directorio con tu nombre (el que mas te guste, no nos des tu DNI) en la unidad d: a ser posible y a partir de ahora utiliza ese directorio para hacer tus prácticas. Ya sabes, subirnos programitas y practicar con ellos :) ... ... ¿cómo? ¿que no sabes crear directorios mediante el CODE/DECODE BUG... repasa los números 2 y tres de Hack x Crack ;p

Puedes crearte tu directorio donde quieras, no es necesario que sea en d:\mellamojuan. Tienes total libertad!!! Una idea es crearlo, por ejemplo, en d:\xxx\system32\default\10019901\mellamojuan (ya irás aprendiendo que cuanto mas oculto mejor :)

Es posiblemente la primera vez que tienes la oportunidad de investigar en un servidor como este sin cometer un delito (nosotros te dejamos y por lo tanto nadie te perseguirá). Aprovecha la oportunidad!!! e investiga mientras dure esta iniciativa (esperemos que muchos años).

- En este momento tenemos mas de 600 carpetas de peña que, como tu, está practicando. Así que haznos caso y crea tu propia carpeta donde trabajar.



MUY IMPORTANTE...

MUY IMPORTANTE!!!!! Por favor, no borres archivos del Servidor si no sabes exactamente lo que estás haciendo ni borres las carpetas de los demás usuarios. Si haces eso, lo único que consigues es que tengamos que reparar el sistema servidor y, mientras tanto, ni tu ni nadie puede disfrutar de él :(Es una tontería intentar "romper" el Servidor, lo hemos puesto para que disfrute todo el mundo sin correr riesgos, para que todo el mundo pueda crearse su carpeta y practicar nuestros ejercicios. En el Servidor no hay ni Warez, ni Programas, ni claves, ni nada de nada que "robar", es un servidor limpio para TI, por lo tanto cuídalo un poquito y montaremos muchos más :)

MANIPULACION DE DOCUMENTOS XML: EL DOM

TERCERA PARTE:

INTERFAZ XMLDOMNode (2)
INTERFAZ XMLDOMNODELIST
INTERFAZ XMLDOMNAMEDNODEMAP
OTRAS INTERFACES DEL DOM

Joaquim Roca Vergés

- DOMNode -- Métodos
- XML + DTD + DOM
- XQL (XML como Base de Datos)

Vimos en el anterior número los atributos de la interfaz DomNode y algunos de sus principales métodos:

- Appendchild = que enganchaba un nodo hijo al final del nodo padre.
- Insertbefore = que enganchaba un nodo hijo dentro del nodo padre, justo antes del nodo hermano (sibling) que indicabamos como referencia.
- Replacenode = que reemplazaba un nodo por otro.
- Removenode = que eliminaba un nodo.
- Clonenode = que hacia una copia de un nodo, que luego teníamos que enganchar con appendchild (tal como vimos en el ejemplo) o con insertbefore (que no vimos en el ejemplo pero que también se puede hacer y os dejo para que probeis)

Vamos a continuar con lo que nos queda de esta interfaz.

INTERFAZ DOMNode –Métodos : Continuación

Hemos visto los métodos para insertar, modificar (reemplazar) y borrar nodos, es decir los métodos con los que podríamos hacer un mantenimiento básico de un xml. Vamos a ver otros métodos interesantes de esta interfaz

MÉTODO HASCHILDNODES

Sintaxis: nodopadre.haschildnodes

Es un método preventivo, informativo. Comprueba si el nodo al que nos referimos tiene hijos.

Devuelve un valor falso si no tiene y verdadero si tiene. Por ejemplo, podemos comprobar si un nodo tiene hijos antes de borrar uno de sus nodos.

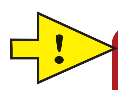
No abrais el Visual basic, os pongo un ejemplo de un trozo de código que realizaría la comprobación:

```
Dim tienehijos As Boolean
'tienehijos tendrá valor true si el nodoPadre
'tiene hijos, de lo contrario tendrá false
tienehijos = nodoPadre.hasChildNodes
'si tiene hijos borramos el nodo
If tienehijos Then
   nodoPadre.removeChild (nodoABorrar)
end if
```

METODOS DE SELECCIÓN DE NODOS: SELECTNODES Y SELECTSINGLENODE (y dos mas por lo bajini)

Sintaxis: nodopadre. SelectNodes(String consulta)

Sintaxis: nodopadre.SelectSingleNode(String consulta)



Un inciso...

Con todo lo que hemos visto hasta ahora, si pensamos un momento, tenemos que podríamos utilizar xml , dtd y dom a modo de una base de datos. No os sorprendáis que en el momento en que os pongáis a trabajar a fondo con el, encontrareis que necesitáis un mecanismo para realizar querys sobre el archivo xml. Utilizando el DOM, podemos acceder a **cualquier** nodo de nuestro documento, pero puede resultaros pesado maniobrar por todas las jerarquías de hijos hasta llegar al nodo en que estáis interesados.

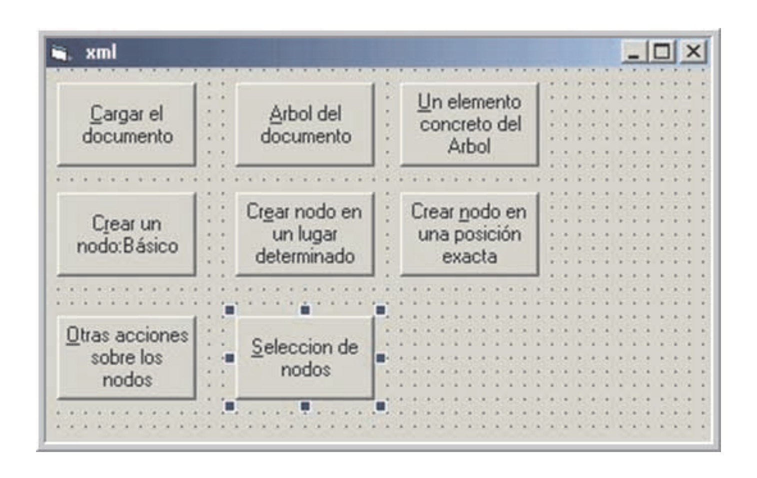
Lo que nos gustaría tener es una versión del SQL para XML, nos gustaría decir "Dame todos los nodos de tipo X, que tengan descendientes de tipo Y" . Y esto existe en XML, y se llama **Xpath** (inicialmente se llamo XQL).

No vamos hablar de Xpath en este curso (de momento), nos ceñiremos al objetivo inicial: XML + DTD +DOM, pero en el caso de que vosotros lo encontréis interesante y siempre que el redactor jefe diga que de acuerdo pues lo tiraremos adelante. Hablar ahora de Xpath, supondría más teoría y pienso que de momento ya habéis tenido bastante, que lo que nos interesa es tocar el DOM y divertirnos.

Bien, pues estos métodos lo que hacen es: uno devuelve el **primer** nodo que cumpla la condición especificada en el argumento **consulta,** el otro devuelve la lista de nodos que cumplen la condición. Vemos estos dos métodos con un ejemplo:

Colocad un botón en el formulario, id a las propiedades y poner:

- **name** = cmdSeleccionNodos
- **caption** = &Seleccion de nodos



Codificad el evento del botón:

```
Private Sub cmdSeleccionNodos_Click()
Dim xmlDocument As DOMDocument
Dim nodoPadre As IXMLDOMNode
Dim unSoloNodo As IXMLDOMNode
Dim listaNodos As IXMLDOMNodeList

Dim xmlString As String
Set xmlDocument = New DOMDocument
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")
'nodoPadre = nodo raiz del documento
Set nodoPadre = xmlDocument.documentElement
MsgBox nodoPadre.xml

' selecciono el primer nodo CLIENTE. El parámetro
' está escrito en Xpath, y es que cuando utilizabais
' el método getelementbytagname, y le pasabais el
```

```
' nombre del nodo, estabais utilizando ya xpath.
' En concreto, le estamos diciendo, seleccióname
' desde el nodo raíz para abajo, en cualquiera de
' los hijos  que tenga, el nodo CLIENTE. Fijaros
' que si el  nodoPadre fuese ORDEN_DE_COMPRA
' en lugar de PEDIDOS, con la instrucción:
' Set nodoPadre = xmlDocument.documentElement.childNodes(0),
' como no encontraría ningún nodo hijo de
' ORDEN_DE_COMPRA cuyos hijos tengan el
' nodo CLIENTE, el programa petaría
Set unSoloNodo = nodoPadre.selectSingleNode("*/CLIENTE")
MsgBox unSoloNodo.xml

' para esta selección, le decimos algo parecido al
' anterior: en concreto le estamos diciendo,
' seleccióname a partir del nodo raíz para abajo,
' en el hijo  ORDEN_DE_COMPRA y solo en ese
' hijo , el nodo CLIENTE
Set unSoloNodo = nodoPadre.selectSingleNode("ORDEN_DE_COMPRA/CLIENTE")
MsgBox unSoloNodo.xml

Set listaNodos = nodoPadre.selectNodes("*/CLIENTE")
Set unSoloNodo = listaNodos.Item(1)
MsgBox unSoloNodo.xml
End Sub
```

Seguro que os preguntáis lo del título de este párrafo "y dos mas por lo bajini" porque no van a haber ejemplos de los dos últimos métodos que quedan de esta interfaz: **TransformNode y transformNodeToObject**.

El motivo es que, lo que hacen estos métodos, es transformar el nodo utilizando una hoja de estilos **XSL.**

Esta hoja de estilos es un archivo externo como puede serlo el dtd o el xml, que se guarda con el nombre: Nombrearchivo.xsl. Y tal como he comentado antes no vamos a hablar de xpath ni de xsl (aunque me gustaría). Si hago el de xpath, vendrá conjuntamente con el de xsl porque están íntimamente ligados.

Finalmente decir que lo que devuelven es una cadena de caracteres el primero y un nodo el segundo con el resultado de la transformación.

INTERFAZ DOMNodeList

Vamos a ver ahora otra interfaz de nodos, la que nos permite trabajar con grupos de nodos. De hecho ya estáis familiarizados con esta interfaz, porque siempre que hemos utilizado getelementsbytagname, el objeto que nos ha devuelto, ha sido un objeto de tipo NodeList.

Una lista de nodos (**nodelist)** es una colección ordenada de nodos. Es el objeto devuelto por todos los métodos y propiedades que pueden devolver más de un nodo como getelementsbytagname y selectnodes.

Los nodos existentes dentro de una lista de nodos, no tienen porque tener nodos hermanos (ej. de nodo hermano = el primer ORDEN_DE_COMPRA es hermano (**sibling)** del segundo ORDEN_DE_COMPRA).

Esta interfaz tiene un solo atributo: **length;** que nos da el número de nodos que tiene la lista de nodos.

Los elementos dentro de NodeList se numeran comenzando por cero, no por uno; esto quiere decir que si tenemos dos elementos en el NodeList, la propiedad length será 2, pero para acceder al primer nodo habrá que usar el índice item(0) y para obtener el segundo se deberá usar item(1).

Una lista de nodos siempre está actualizada, es decir, si añadimos o eliminamos nodos del documento, la lista reflejará inmediatamente estos cambios.

Por ejemplo, si añadiésemos un tercer nodo ORDEN_DE_COMPRA, la lista se actualizaría sin tener que hacer nada, inmediatamente después de añadirlo, podríamos preguntar por item(2) y si preguntásemos por la length de la lista, en lugar de devolvernos 2 nos devolvería 3.

Esta interfaz tiene tres métodos:

1. item = que nos permite acceder a los elementos de la lista, a los nodos de la lista, teniendole que indicar la posición del nodo. Ejemplo:

```
Dim xmlDocument As DOMDocument
Dim listadenodos As IXMLDOMNodeList
Dim nodo As IXMLDOMNode
Set xmlDocument = New DOMDocument
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")
Set listadenodos =xmlDocument.getElementsByTagName("ORDEN_DE_COMPRA")

' con listadenodos(0) is Nothing y con IsNull(listadenodos(0)) podemos
' comprobar que  listadenodos tiene nodos del tipo que le hemos
' indicado. Si no tiene, salimos de la subrutina.
If listadenodos(0) Is Nothing Or IsNull(listadenodos(0)) Then
   Exit Sub
End If
Set nodo = listadenodos.Item(0)
MsgBox nodo.Text
'nodo tendría el primer ORDEN_DE_COMPRA del xml
```

2. nextNode= que nos devuelve el siguiente nodo de la lista. Ejemplo:

```
Set nodo= listadenodos.nextNode
MsgBox nodo.Text
' nodo tendría ahora el segundo
' ORDEN_DE_COMPRA del xml
```

3. reset= inicializa la posición de la lista de nodos justo antes de la primera posición, justo antes del primer nodo. Ejemplo:

```
' estábamos en el segundo ORDEN_DE_COMPRA
' del xml y hacemos un reset
' ahora estamos justo antes de el primer nodo
listadenodos.Reset
'al hacer un nextNode, nos devuelve el siguiente
' nodo, que es el primero ya que estábamos justo
' antes de este.
Set nodo = listadenodos.nextNode
```

INTERFAZ DOMNamedNodeMap

Los nodos que se suelen tratar más comúnmente con esta interfaz son los nodos atributos.

Recordemos que los atributos son nodos.

Esta interfaz es parecida a la interfaz Node, con la excepción de que te deja referenciar los nodos por nombre o por el número. Es igual a un array hash o a una colección.

¿Qué es un **hash array**? Pues un array asociativo, jajaja. Y un array asociativo es un array donde el índice es la clave y el contenido es el valor, jajaja.

Lo siento, cada vez que intento aclararlo lo releo y veo que lo he puesto mas complicado. Lo vemos como siempre con un ejemplo.

¿Como hacíamos hasta ahora para referenciar un nodo? Le dábamos un número, le decíamos que *setnode = listadenodos.Item(0)* o bien *Set nodoPadre = xmlDocument.documentElement.childNodes(0)*, u otras maneras. En cualquier caso siempre colocábamos un número, para decirle qué nodo, para referenciar el nodo.

Ahora bien, en un hash array, en lugar de colocar un número, colocamos un nombre, un nombre que identifica al nodo, y asociado a ese nombre (a esa clave con la que identificamos al nodo, a ese índice con el que identificamos al nodo) tenemos su valor. En lugar de decir item(0) diremos item("a").

Si tuviéramos : <ORDEN_DE_COMPRA num="123">, nos referiríamos a su atributo num con la instrucción **getnameditem("num").nodevalue,** y ese par num –valor de num- es lo que es la hash table.



Esta interfaz...

Esta interfaz contiene únicamente nodos de tipo NOTATION, ENTITY o ATTR.
Aunque a partir de ahora pueda referirme con la expresión "lista de nodos atributo" porque es lo más usual, porque es lo que ocurre en el 95% de los casos, todo lo que explique referente a una lista de nodos atributo, también es aplicable a una lista de nodos entity o a una lista de nodos notation.

Para practicar con esta interfaz, vamos a modificar el dtd y el xml. Vamos a añadirle a ORDEN_DE_COMPRA un atributo que llamaremos **num** que será de tipo ID(recordamos que un atributo de tipo ID, es un atributo único, cada uno de los ORDEN_DE_COMPRA que tengamos tendrán un atributo num único) y requerido, y esto lo especificaremos con la palabra clave #REQUIRED.

Vamos también a añadir un atributo **dni** a CLIENTE, también de tipo ID, pero esta vez le vamos a decir que es #IMPLIED (opcional), dicho de otro modo, vamos a decirle que el cliente puede o no puede proporcionarnos el dato del dni, pero si nos lo proporciona, debe de ser único entre todos los CLIENTE del documento xml.

¿Ya no os acordabais? Si no os acordabais, repasad por favor el PC Paso a Paso número 12.

De modo que el dtd pedidos quedará de la siguiente manera:

```
<!ELEMENT PEDIDOS (ORDEN_DE_COMPRA+)>
<!ELEMENT ORDEN_DE_COMPRA (CLIENTE, PRODUCTO+)>
<!ATTLIST ORDEN_DE_COMPRA num ID #REQUIRED>
<!ELEMENT PRODUCTO (#PCDATA)>
<!ELEMENT CLIENTE (NUMERO_DE_CUENTA, NOMBRE_COMPLETO)>
<!ATTLIST CLIENTE  DNI  ID #IMPLIED>
<!ELEMENT NUMERO_DE_CUENTA (#PCDATA)>
<!ELEMENT NOMBRE_COMPLETO (NOMBRE, APELLIDO1, APELLIDO2)>
<!ELEMENT NOMBRE (#PCDATA)>
<!ELEMENT APELLIDO1 (#PCDATA)>
<!ELEMENT APELLIDO2 (#PCDATA)>
```

y por lo tanto, deberemos cambiar pedidos.xml, para proporcionarle a todos los ORDEN_DE_COMPRA su **num** de tipo ID.

Atención que los atributos ID deben cumplir las reglas de nombres XML, esto es que **no pueden comenzar por un número.** Y como lo que queremos es darle un número, antepondremos un carácter válido, como puede ser el **underscore ("_").** Pedidos.xml quedará de la siguiente forma:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE PEDIDOS SYSTEM "pedidos.dtd">
<PEDIDOS>
  <ORDEN_DE_COMPRA num="_01">
    <CLIENTE>

  <NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
      <NOMBRE_COMPLETO>
        <NOMBRE>Sam</NOMBRE>
        <APELLIDO1>Bass</APELLIDO1>
        <APELLIDO2/>
      </NOMBRE_COMPLETO>
    </CLIENTE>
    <PRODUCTO>LIBRO</PRODUCTO>
  </ORDEN_DE_COMPRA>
  <ORDEN_DE_COMPRA num="_02">
    <CLIENTE DNI="J5555555">

  <NUMERO_DE_CUENTA>987654321</NUMERO_DE_CUENTA>
      <NOMBRE_COMPLETO>
        <NOMBRE>Jesse</NOMBRE>
        <APELLIDO1>James</APELLIDO1>
        <APELLIDO2/>
      </NOMBRE_COMPLETO>
    </CLIENTE>
    <PRODUCTO>DISCO DE VINILO</PRODUCTO>
    <PRODUCTO>SUPERSNAZZ - FLAMIN' GROOVIES</PRODUCTO>
  </ORDEN_DE_COMPRA>
</PEDIDOS>
```

MÉTODOS DE LA INTERFAZ DOMNamedNodeMap

Esta interfaz solo tiene un atributo: **length,** que nos devuelve el número de atributos que contiene la lista de atributos.

La lista de atributos que contiene un objeto de tipo DOMNodeMap, hemos dicho al principio que es muy parecida a la interfaz nodelist y donde nodelist tenía una lista de nodos, namednodemap tiene una lista de atributos de un nodo. Y repito por favor que tengáis en mente que un atributo es un nodo a su vez. Por ejemplo:

ORDEN_DE_COMPRA tiene una lista de nodos atributos de una **length** de un elemento. Es decir que el número de elementos que tiene es = 1. Si preguntáramos por la length de ORDEN_DE_COMPRA, nos devolvería un 1.

Vamos a analizar los métodos genéricos de esta interfaz.

MÉTODOS GetNamedItem, SetNamedItem y RemoveNamedItem

Sintaxis getNamedItem(String strNombre)

GetNamedItem devuelve una referencia al nodo que tiene el nobre **strNombre,** el parámetro que le pasamos a la función. Es decir, que una vez ejecutada la función, habrá una variable de tipo **node** que tendrá una referencia al nodo atributo que devuelve la función. No os preocupéis si no lo habéis entendido, mirad el ejemplo y ya veréis que sencillo es. Pero luego volved a leer la teoría eeeeeeeeeh!!

Sintaxis SetNamedItem (Node NodoArgumento)

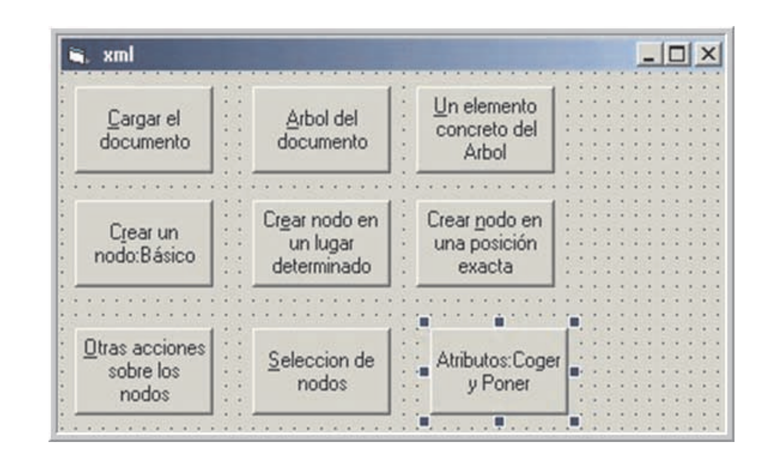
SetNamedItem es un método que nos permite añadir un nodo a la lista de nodos atributos(o nodos entidades o nodos notation) de la interfaz DOMNamedNodeMap. Si hay un nodo que tiene el mismo nombre, es reemplazado**.** Devuelve una referencia al nuevo nodo, si estamos añadiendo un nuevo nodo atributo, y una referencia al nodo que ha reemplazado al antiguo nodo en caso de que haya un reemplazo de nodos.

Sintaxis removeNamedItem(String strNombre)

removeNamedItem es un método qu borra el nodo que especificamos en el parámetro **strnombre.** En el caso que intentáramos borrar un nodo de tipo Atributo, y el nodo

tuviera un valor por defecto (valor que deberemos haber definido en nuestro DTD), este nodo será reemplazado por otro con el valor por defecto de manera automática.

Vemos estos tres métodos con un ejemplo:



Colocad un botón en el formulario, id a las propiedades y poner:

- **name** = cmdCogerPonerAtributos
- **caption** = Atributos:Coger y Poner

Codificad el evento del botón:

```
Private Sub cmdCogerPonerAtributos_Click()

Dim xmlDocument As DOMDocument
Dim nodoAtributo As IXMLDOMNode
Dim listaNodos As IXMLDOMNodeList
Dim listaAtributos As IXMLDOMNamedNodeMap
Dim nuevoNodoAtributo As IXMLDOMNode

Set xmlDocument = New DOMDocument
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")
MsgBox xmlDocument.documentElement.childNodes(0).xml

'atributos tendrá todos los nods atributos del segundo nodo del
'árbol xml. Y tendrá el segundo porque documentElement nos
'devuelve el elemento raíz que es PEDIDOS.
'Por tanto childNodes (0) = tiene el primer
'ORDEN_DE_COMPRA, y atributos la lista de los nodos atributos
'que tiene el primer 'ORDEN_DE_COMPRA
Set listaAtributos = xmlDocument.documentElement.childNodes(0).Attributes

'mostramos un mensaje con el número de nodos atributos "num",
'y efectivamente, tal como hemos explicado en la teoría, el visual
'basic nos dice que tenemos un atributo "num" en ese nodo. Volved
'a repasar la teoría!!
MsgBox listaAtributos.length

'referenciamos el nodo atributo "num", de manera que a partir de
'este momento, nodoAtributo lo tendrá referenciado
Set nodoAtributo = listaAtributos.getNamedItem("num")
MsgBox nodoAtributo.nodeValue

'como nodeValue devuelve el valor del nodo también podríamos
'hacer: MsgBox atributos.getNamedItem("num").nodeValue

'le cambiamos el valor al atributo y lo visualizamos con un mensaje
listaAtributos.getNamedItem("num").Text = "_03"
MsgBox listaAtributos.getNamedItem("num").nodeValue

'creamos un nodo de tipo Atributo y le asignamos contenido
Set nuevoNodoAtributo = xmlDocument.createNode(NODE_ATTRIBUTE, "Vendedor", "")
nuevoNodoAtributo.Text = "Manel of the Bomb"

'esto también lo podríamos haber hecho con la interfaz
'DomAttribute de la siguiente manera:
'Dim atributo As IXMLDOMAttribute
'Recordad que aunque hemos visto las principales, el DOM
'tiene más interfaces. Habíamos visto en otros números la interfaz
'DOMElement, ahora podéis ver que existe la DOMAttribute...
'efectivamente, para cada tipo de nodo hay una interfaz DOM,
'así tenemos DOMComment, DOMCDATASection etc.
'Set atributo = xmlDocument.createAttribute("Vendedor")
'atributo.nodeValue = "Manel of the bomb"
'Set nuevoNodoAtributo = listaAtributos.setNamedItem(atributo)

'le decimos a la listaAtributos que añada el nuevo atributo, y lo
'referenciamos a la variable de tipo nodo nuevoNodoAtributo
Set nuevoNodoAtributo = listaAtributos.setNamedItem(nuevoNodoAtributo)

'vemos que la longitud de la lista de atributos pasa a ser de uno
'a dos:
'ahora además de "num" tenemos el atributo "vendedor"
MsgBox listaAtributos.length

'comprobamos que hemos añadido el nuevo atributo en el lugar
'correcto
MsgBox xmlDocument.documentElement.childNodes(0).xml

'si creamos un nodo atributo con un nombre ya existente lo
'reemplaza.
'lo comprobamos con un mensaje
Set nuevoNodoAtributo = xmlDocument.createNode(NODE_ATTRIBUTE, "Vendedor", "")
nuevoNodoAtributo.Text = "Bombing of the manel"
Set nuevoNodoAtributo = listaAtributos.setNamedItem(nuevoNodoAtributo)
MsgBox xmlDocument.documentElement.childNodes(0).xml

'Finalmente Borramos el nodo atributo que habiamos creado
listaAtributos.removeNamedItem ("Vendedor")
MsgBox xmlDocument.documentElement.childNodes(0).xml

'¿Qué ocurriría si borrásemos el atributo "num"? Si no lo
'validásemos contra un DTD, no ocurriría nada en absoluto, si
'lo validásemos contra un DTD, el xml petaría porque recordemos
'que "num" lo hemos definido como un tipo ID. Y un tipo ID es
'siempre de existencia Obligatoria.

End Sub
```

OTRAS INTERFACES DEL DOM

Hemos visto las cuatro interfaces básicas del DOM y hemos comentado que habían algunas más, hemos comentado que cada uno de los tipos de nodos que existen tienen su propia interfaz, podría extenderme con cada una de ellas pero no lo haré porque pienso que sería repetir lo que ya hemos aprendido, creo de corazón que si no tenéis problemas con las otras interfaces no vais a tener ningún problema con las interfaces de nodos y que básicamente son lo mismo que la interfaz DOMNode, con algún que otro método particular a la interfaz.

No obstante, me gustaría comentar la interfaz de nodos **DocumentFragment** porque es un poco especial, y quiero que veais un ejemplo.

Finalmente y para finalizar las interfaces del DOM hablaremos de DOMException, que es la interfaz que sirve para gestionar los errores.

A) DOCUMENT FRAGMENT

Tal como explicamos en el artículo de la teoría del dom, esta interfaz es un trozo de xml que está suelto en el documento.

Un documento XML solo puede tener un elemento raiz, y en nuestro xml es PEDIDOS, pero nos podría resultar de mucha utilidad tener en un lugar temporal un trozo de xml, independientemente de que esté o no bien formado.

Imaginemos que vamos a crear un nuevo nodo NOMBRE_COMPLETO con todos sus nodos hijos. Pues nos vendría bien tener en un lugar temporal los nodos mientras los vamos añadiendo hasta que formen una estructura de nodos igual a :

```
<NOMBRE_COMPLETO>
        <NOMBRE>Wild</NOMBRE>
        <APELLIDO1>Bill</APELLIDO1>
        <APELLIDO2>Hichtcock</APELLIDO2>
</NOMBRE_COMPLETO>
```

o también podríamos tener un programa con la opción de cortar y pegar un nodo. Pues bien para hacer ambas cosas podríamos utilizar la interfaz documentfragment.

Ya que vamos a añadir un nodo hijo NOMBRE_COMPLETO a CLIENTE, vamos a tocar primero el DTD, de manera que nos permita más de un NOMBRE_COMPLETO por CLIENTE. Lo hacemos añadiendo un signo + a NOMBRE_COMPLETO de modo que esa línea en el DTD quede así:

```
<!ELEMENT CLIENTE (NUMERO_DE_CUENTA, NOMBRE_COMPLETO+)>
```

Añadimos un NOMBRE COMPLETO al CLIENTE del segundo ORDEN_DE_COMPRA, y lo validamos contra el DTD con la herramienta que os recomendé que os bajarais, el xmlspy (http://www.altova.com/download.html) .

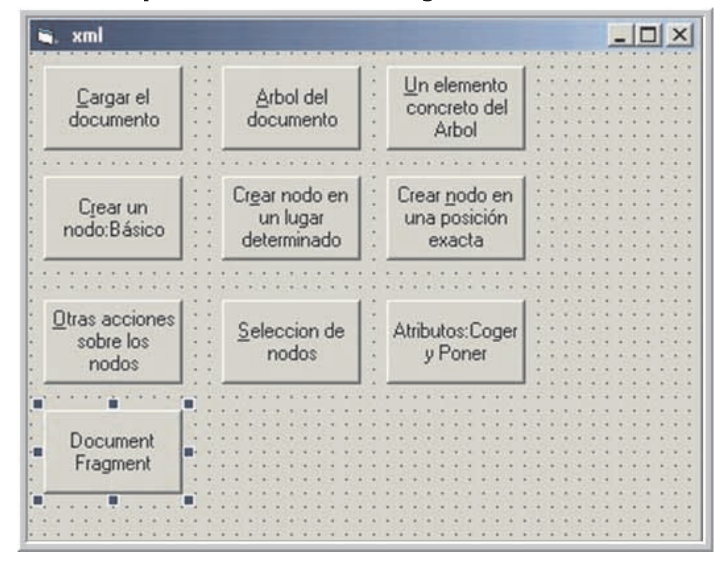
El segundo nodo ORDEN_DE_COMPRA de pedidos.xml debe de quedar así:

```
<ORDEN_DE_COMPRA num="_02">
   <CLIENTE DNI="J5555555">
      <NUMERO_DE_CUENTA>987654321</NUMERO_DE_CUENTA>
      <NOMBRE_COMPLETO>
         <NOMBRE>Jesse</NOMBRE>
         <APELLIDO1>James</APELLIDO1>
         <APELLIDO2/>
      </NOMBRE_COMPLETO>
      <NOMBRE_COMPLETO>
         <NOMBRE>Julius</NOMBRE>
         <APELLIDO1>James</APELLIDO1>
         <APELLIDO2>joffa</APELLIDO2>
      </NOMBRE_COMPLETO>
   </CLIENTE>
   <PRODUCTO>DISCO DE VINILO</PRODUCTO>
   <PRODUCTO>SUPERSNAZZ - FLAMIN'GROOVIES</PRODUCTO>
</ORDEN_DE_COMPRA>
```

Lo mejor como siempre un ejemplo, y luego volver a la teoría.

Colocad un botón en el formulario, id a las propiedades y poner:

- **name** = cmdDocumentFragment
- **caption** = Document Fragment



Codificad el evento del botón:

```
Private Sub cmdDocumentFragment_Click()
Dim xmlDocument As DOMDocument
Dim fragmentoXML As IXMLDOMDocumentFragment
Dim nodoRaiz As IXMLDOMNode
Dim nodoHijo As IXMLDOMNode
Set xmlDocument = New DOMDocument
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")

'creamos la interfaz documentFragment
Set fragmentoXML = xmlDocument.createDocumentFragment

'el nodo raíz va a ser NOMBRE_COMPLETO,  al que le vamos
'a añadir NOMBRE, APELLIDO1 y APELLIDO2
Set nodoRaiz = xmlDocument.createNode(NODE_ELEMENT, "NOMBRE_COMPLETO", "")

'el fragmento xml consta ahora de un solo elemento:
'NOMBRE_COMPLETO, y como no tiene contenido,   se representa
'como <NOMBRE_COMPLETO/>
Set nodoRaiz = fragmentoXML.appendChild(nodoRaiz)
Msgbox nodoRaiz.xml

'le añadimos los nodos hijos, añadimos texto a los Nodos Elementos,
'Añadimos el elemento con su texto al nodo raíz, y finalmente
'añadimos todo el nodo raíz al fragmento xml
Set nodoHijo = xmlDocument.createNode(NODE_ELEMENT, "NOMBRE", "")
nodoHijo.Text = "Wild"
Set nodoHijo = nodoRaiz.appendChild(nodoHijo)
Set nodoHijo = xmlDocument.createNode(NODE_ELEMENT, "APELLIDO1", "")
nodoHijo.Text = "Bill"
Set nodoHijo = nodoRaiz.appendChild(nodoHijo)
Set nodoHijo = xmlDocument.createNode(NODE_ELEMENT, "APELLIDO2", "")
nodoHijo.Text = "Hichtcock"
Set nodoHijo = nodoRaiz.appendChild(nodoHijo)

'ahora el nodo hijo tiene todo el fragmento xml
Set nodoHijo = fragmentoXML.appendChild(nodoRaiz)

'el siguiente mensaje nos mostrará el fragmento xml, :
'<NOMBRE_COMPLETO>
'<NOMBRE>Wild</NOMBRE>
'<APELLIDO1>Bill</APELLIDO1>
'<APELLIDO2>Hichtcock</APELLIDO2>
'</NOMBRE_COMPLETO>
MsgBox fragmentoXML.xml

'nodohijo también tiene todo el fragmento xml, podéis ver que os
'voy poniendo
'combinaciones, para que no os quedéis siempre con el mismo
'código
MsgBox nodoHijo.xml

' finalmente lo añadimos al xml
' ya que nodohijo y fragmentoxml tienen el mismo nodo, sería
' correcta cualquiera de las dos siguientes sentencias
' xmlDocument.documentElement.childNodes(1).childNodes(0).appendChild nodoHijo
xmlDocument.documentElement.childNodes(1).childNodes(0).appendChild fragmentoXML

'mostramos un mensaje de cómo ha quedado el documento después
'de añadir el fragmento y vemos que efectivamente se ha añadido
MsgBox xmlDocument.documentElement.xml

'ahora el nodo CLIENTES, del segundo ORDEN_DE_COMPRA
'tiene 3 elementos y el que hemos añadido es el número tres, lo
'comprobamos (recordad que se empieza a numerar por 0, por lo
'tanto sera childnodes(2) en lugar de childnodes(3)
MsgBox
xmlDocument.documentElement.childNodes(1).childNodes(0).childNodes(2).xml
End Sub
```

¿es todo? Si, pero, ¿no hay algo en este código que no está bien? ¿No hay algo en este código que hace que tengáis dolor en los ojos, algo dentro vuestro que os dice que hay algo que no marcha?

Efectivamente, fijaros cuantas veces hemos escrito:

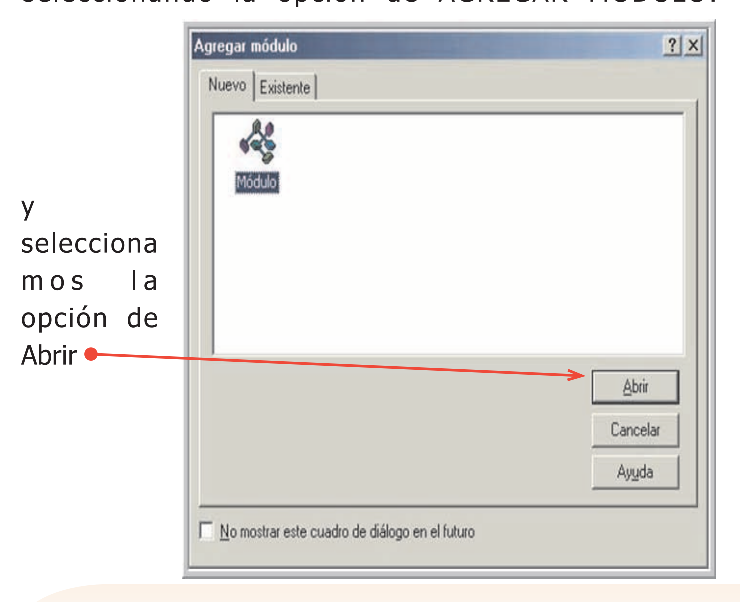
```
Set nodoHijo = xmlDocument.createNode(NODE_ELEMENT, "NOMBRE", "")
nodoHijo.Text = "Wild"
Set nodoHijo = nodoRaiz.appendChild(nodoHijo)
```

Y si nos fijamos solo cambia el nombre del nuevo nodo y el texto que tendrá. Uno de los objetivos de todo buen programador es no repetir ni una línea de código, y a ser posible programar de manera que el código que escribamos sea reutilizable para otro programa.

Otra cosa es la realidad claro, empiezan los proyectos en un mundo maravilloso, pero a la hora de entregar, todo son prisas y a menudo terminamos haciendo cortar y pegar por todas partes, y repitiendo código a mansalva. Pero ya que tenemos un poco de tiempo y no hay proyecto que nos presione, vamos a ver como pasar por parámetro y como recibir como resultado, los nodos del DOM XML

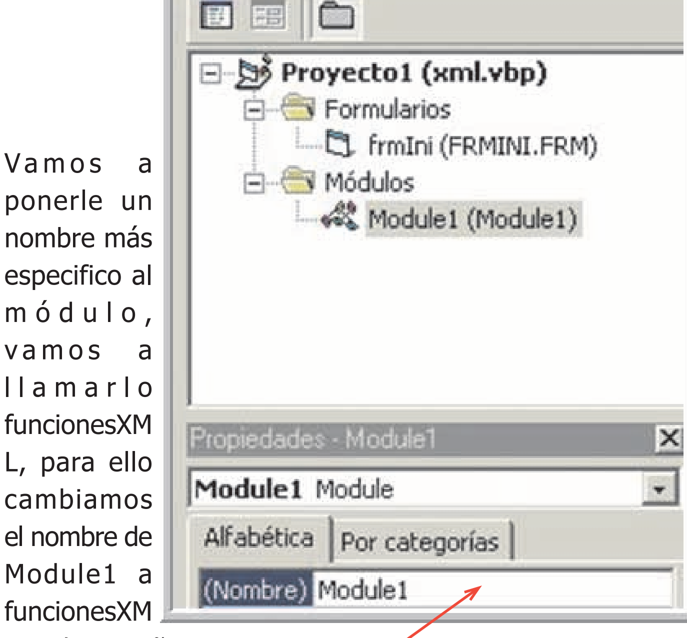
Así pues, efectivamente, aquí vendría una función , vamos a hacer una función que añada un nodo hijo a un nodo padre. Y la vamos a incluir en un módulo, de manera que en un futuro podamos incluirlo en cualquier otro proyecto.

Volvemos al Visual Basic y vamos al menú PROYECTO, seleccionando la opción de AGREGAR MODULO:



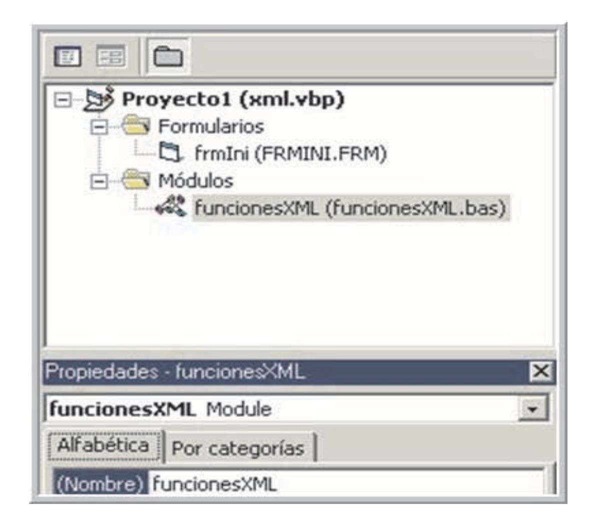
y seleccionamos la opción de Abrir

En el explorador de proyectos, vemos que se nos ha añadido una carpeta con nombre “Módulos” y un archivo que se llama Module1.



Vamos a ponerle un nombre más especifico al módulo, vamos a llamarlo funcionesXML, para ello cambiamos el nombre de Module1 a funcionesXML en la pestaña de propiedades.

Despues haremos click en el botón de guardar, y cuando os pregunte por el nombre, le indicáis que se llamará funcionesXML, de manera que el explorador de proyectos os quede de la siguiente manera:



Abrid el módulo y escribid lo siguiente:

```
Sub mensajeXML(nodo As IXMLDOMNode)
'recibe un nodo y muestra un mensaje, con el xml del nodo
    MsgBox nodo.xml
End Sub

Function crearNodo(doc As DOMDocument, nodo As IXMLDOMNode, tipoNodo As Integer, nombreNodo As String) As IXMLDOMNode

'creamos el nodo, puede parecer más trabajoso, pero pensad
'que si tenéis que hacer una modificación, con cambiar aquí hemos
'modificado en todo el proyecto
Set nodo = doc.createNode(tipoNodo, nombreNodo, "")
'como lo que devolvemos es un objeto, la función debe hacer un set
Set crearNodo = nodo

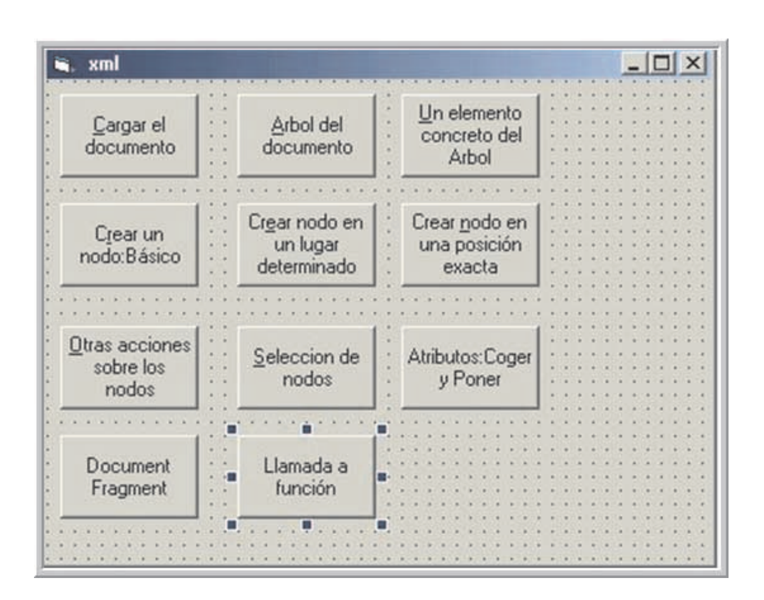
End Function

Function apenNodo(nodo As IXMLDOMNode, nodoDondePegamos As IXMLDOMNode) As IXMLDOMNode

'fijaros que podéis hacerlo de una sola vez, aunque es menos
'fácil de leer, os ahorra tiempo, comparadlo con la función
'crearNodo
Set apenNodo = nodoDondePegamos.appendChild(nodo)

End Function
```

Volved al formulario, colocad un botón:



id a las propiedades y escribid:

- **name** = cmdFuncion
- **caption** = Llamada a función

Codificad el evento del botón:

```
Private Sub cmdFuncion_Click()
Dim xmlDocument As DOMDocument
Dim fragmentoXML As IXMLDOMDocumentFragment
Dim nodoRaiz As IXMLDOMNode
Dim nodoHijo As IXMLDOMNode
Dim pegarNodo As IXMLDOMNode

Set xmlDocument = New DOMDocument
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")

Set fragmentoXML = xmlDocument.createDocumentFragment

Set nodoRaiz = crearNodo(xmlDocument, nodoRaiz, NODE_ELEMENT, "NOMBRE_COMPLETO")

Set nodoRaiz = apenNodo(nodoRaiz, fragmentoXML)
'llamamos a la subrutina mensajeXML, pasando como párametro
'un nodo como una subrutina no devuelve nada, escribimos
'nombreSubrutina parametro
'si fuera una función escribiríamos
'variable=nombrefuncion(parametro)
mensajeXML nodoRaiz
Set nodoHijo = crearNodo(xmlDocument, nodoHijo, NODE_ELEMENT, "NOMBRE")
nodoHijo.Text = "Wild"
Set nodoHijo = apenNodo(nodoHijo, nodoRaiz)
Set nodoHijo = crearNodo(xmlDocument, nodoHijo, NODE_ELEMENT, "APELLIDO1")
nodoHijo.Text = "Bill"
Set nodoHijo = apenNodo(nodoHijo, nodoRaiz)
Set nodoHijo = crearNodo(xmlDocument, nodoHijo, NODE_ELEMENT, "APELLIDO2")
nodoHijo.Text = "Hichtcock"
Set nodoHijo = apenNodo(nodoHijo, nodoRaiz)
Set nodoHijo = fragmentoXML.appendChild(nodoRaiz)
'la función mensajeXML espera un nodo, fragmentoXML es un nodo
mensajeXML fragmentoXML
mensajeXML nodoHijo
xmlDocument.documentElement.childNodes(1).childNodes(0).appendChild fragmentoXML

'la función mensajeXML espera un nodo, documentElement es un nodo
mensajeXML xmlDocument.documentElement
```

```
' evidentemente, los childNodes de documentElement
' son también nodos
mensajeXML xmlDocument.documentElement.childNodes(1).childNodes(0).childNodes(2)

End Sub
```

B) DOMEXCEPTION

Para finalizar, comentaremos la interfaz DOMException, que es la interfaz que gestiona los errores que devuelve el DOM

Todas las operaciones que ejecutéis utilizando el DOM, que no estén permitidas deberían lanzar un error (por ejemplo un determinado tipo de nodo no admite la operación que estáis realizando), pero Microsoft no lanza este error, por tanto en el DOM de Visual Basic con el que estáis programando no existe esta interfaz. Sabed sin embargo que debería de existir y que en otros lenguajes (en java por ejemplo) si existe.

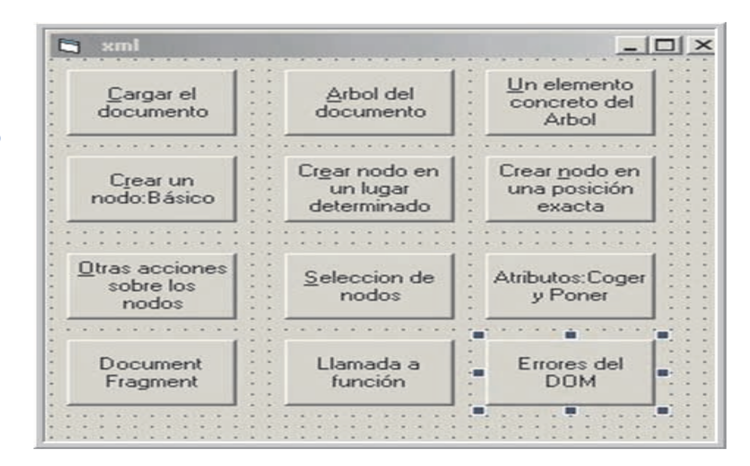
Lo que proporciona Microsoft es una propiedad, dentro de la interfaz DOMDocument que se llama **parseError,** y esta propiedad a su vez tiene un listado de propiedades, por ejemplo: parseError.**errorCode** nos devolverá el código del error, y parseError.**reason** la descripción del error.

Lo vemos con un ejemplo en el que generaremos un error de carga del xml, y lanzaremos un mensaje con todas las propiedades de **parseError** y su explicación.

Vamos al formulario y ponemos un botón

En las propiedades y escribid:

name = cmdErrores
· **caption** = Errores del DOM



Codificad el evento click del botón:

```
Private Sub cmdErrores_Click()
Dim xmlDocument As DOMDocument
Dim strString As String
Dim blnValido As Boolean
Set xmlDocument = New DOMDocument

xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True

blnValido = xmlDocument.Load("C:\xmlDom\Compraxx.xml")
If blnValido Then
   MsgBox "El documento es válido"
Else

'1- parseError.errorCode

strString = " El código de error es " & xmlDocument.parse.Error.errorCode & Chr(13)

'2- parseError.filePos

strString = strString & " La posición absoluta del archivo donde ha ocurrido el
error es  " & xmlDocument.parseError.filepos & Chr(13)

'3- parseError.Line

'Si el error se hubiera producido en una línea determinada del
'archivo xml, se nos indicaria aquí
strString = strString & " Ha ocurrido en la línea  " & xmlDocument.parseError.Line
& " como no ha cargado el documento, y ahi es donde ha dado el error, la linia
es la cero" & Chr(13)
```

```
'4- parseError.linepos

strString = strString & " Dentro de la linea anterior, el error se ha generado a partir del carácter nº " & xmlDocument.parseError.linepos & Chr(13)

'5- parseError.reason

strString = strString & " La descripción del error es "  & xmlDocument.parseError.reason & Chr(13)

'6- parseError.srcText

' como no ha cargado el documento xml (nos ha dado un error de ' carga), srcText=""
'es decir, no tiene valor, lo veréis en el mensaje del final
strString = strString & " El texto entero de la línea que contenia el error es " & xmlDocument.parseError.srcText & Chr(13)

'7- parseError.url

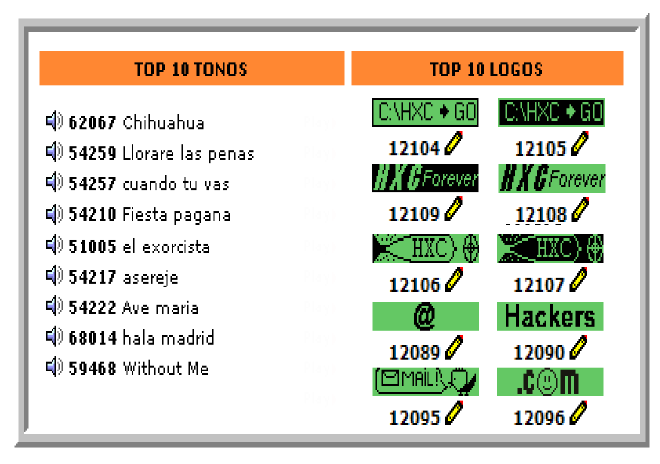
strString = strString &
   " El documento que ha dado el error esta ubicado en " &
   xmlDocument.parseError.url
   MsgBox strString
Exit Sub
End If
On Error Resume Next
Set xmlDocument = Nothing
End Sub
```

Espero que os haya gustado y sobretodo que hayáis entendido bien tanto XML, DTD como DOM. Si lo habéis comprendido y no os ha quedado ninguna duda, tened en cuenta que lo hemos dado todo, y que esos tres palos del xml los domináis a la perfección. Me gustaría continuar con XSL y Xpath, igual dentro de un par de meses. Hasta entonces...

¡Saludos compañeros!



Escribe un mensaje con el texto : PCLOG + el código del logo ó melodia + la marca de tu móvil y envíalo al 7227



HAY MUCHOS MAS EN
http://pclog.buscalogos.com/

EL GANADOR DEL SORTEO DE UN SUSE LINUX 9 DEL MES DE DICIEMBRE ES:

JOSE CARLOS GARCIA BLANCO

TARRAGONA

SEGUIR LLAMANDO, EL PROXIMO PODRIA SER PARA TI (PAG 42)

¿QUIERES COLABORAR CON PC PASO A PASO?

PC PASO A PASO busca personas que posean conocimientos de informática y deseen publicar sus trabajos.

SABEMOS que muchas personas (quizás tu eres una de ellas) han creado textos y cursos para "consumo propio" o "de unos pocos".

SABEMOS que muchas personas tienen inquietudes periodísticas pero nunca se han atrevido a presentar sus trabajos a una editorial.

SABEMOS que hay verdaderas "obras de arte" creadas por personas como tu o yo y que nunca verán la luz.

PC PASO A PASO desea contactar contigo!

NOSOTROS PODEMOS PUBLICAR TU OBRA!!!

SI DESEAS MÁS INFORMACIÓN, envíanos un mail a empleo@editotrans.com y te responderemos concretando nuestra oferta.

SERIE XBOX LIFE
EVOLUTION X TU MEJOR AMIGO

- Aprenderemos qué es y cómo se instala Evolution X.
- Descubriremos el Servidor FTP de la XBOX :)
- Haremos Copias de Seguridad de nuestros juegos

Muy buenas a todos todas :)

En este artículo, en primer lugar veremos qué es y cómo se instala el famoso Evolution X, en segundo lugar veremos cómo hacer una copia de seguridad de un juego.

! Sobre las copias...

Sobre las copias de seguridad de tus juegos: Imaginemos que compramos un juego de la XBOX y a nuestro perro, un buen día, le da por mordisquearlo pensando que es un Donette. Se acabó, adiós a nuestro juego. PUES NO!!

Aparentemente no se pueden hacer copias (de seguridad) de los juegos de la XBOX, pero nosotros vamos a iniciarte en esta saludable práctica (y el mes que viene profundizaremos en ello).

Hay quienes comparten sus copias de seguridad de los juegos de la XBOX en Internet (generalmente con el e-mule, www.spanishare.com). Allá cada uno con su conciencia :)

Lo que os voy a explicar en un principio puede parecer complicado, pero como veréis mas adelante, no lo es tanto.

Antes de empezar, vamos a dejar muy claritas unas cuantas cosas.

ACLARANDO CONCEPTOS: La Autista-Implicito-Patia

Los que "creamos" esta revista tenemos una enfermedad MUY DIFÍCIL de erradicar. Creemos que todo el mundo sabe "lo que se mueve" en el "mundillo", creemos que todo el mundo ha oído hablar de los Mod Chip y que, quien mas / quien menos, todos sabemos descargar juegos de Internet para la XBOX y jugar a ellos.

CRASO ERROR!!! Hemos recibido cientos de e-mails de personas que no tienen ni idea de lo que es un Mod Chip y mucho menos de grabar juegos bajados de Internet para la XBOX. ¿INCREÍBLE? No, pura realidad.

Los que estamos todo el día liados con ordenadores y extraños textos llenos de incomprensibles símbolos, poco a poco desarrollamos una terrible enfermedad: La Autista-implicito-patia. No, los psicólogos no la conocen, pero los "pillados" por la Informática la padecemos y muchas veces ni nos enteramos.

Implícito porque, cuando hablamos de algo, creemos que todos saben de qué va el tema y utilizamos infinidad de conceptos técnicos que quizás muy pocos conocen en profundidad.

Autista porque ni siquiera somos capaces darnos cuenta cuando alguien no tiene ni idea de lo que le estamos contando.

Vamos a ver si podemos curarnos de esa enfermedad :)

Aclarando Conceptos: Un repaso al "mundillo" de la XBOX.

La XBOX es la consola de videojuegos de Microsoft (vale, vale, me he lucido).

Superada la vergüenza, vamos a lo importante. Las consolas de hoy en día tienen una serie de sistemas de seguridad que, entre otras cosas y hablando de la XBOX:

- te impiden ejecutar juegos que no sean originales (no puedes hacer copias de seguridad de tus juegos)

- te impide visualizar películas en DVD que no son de la Zona correspondiente (en este caso España)

- te impide ejecutar software que no esté "firmado" por Microsoft. Es decir, que si quieres programar algo para la XBOX debes obtener la conformidad del tio Hill y por supuesto PAGAR por ella.

En resumen, que compras un producto totalmente "capado", podría hacer muchas más cosas PERO Microsoft no te deja. Es como comprar un ordenador que únicamente te permitiese ejecutar productos de Microsoft (o del fabricante de turno).

Aquí empieza "la guerra". La gente como nosotros queremos sacarle el mayor partido a nuestras compras y, si alguien nos pone límites, pues nosotros los rompemos. Queremos poder visualizar DVDs de cualquier país del mundo, queremos poder hacer copias de seguridad de nuestros juegos (por si se nos estropean), queremos ejecutar programas libremente (sean o no certificados por Microsoft), en definitiva, no queremos ataduras.

Aclarando Conceptos: Los Mod Chips

¿Cómo podemos hacer todo eso? Pues bien, hay una serie de señores que han fabricado unos circuitos integrados que, una vez instalados en el interior de la consola (hay que soldarlos), nos permiten hacer todo eso que antes no podíamos.

En el número anterior de la revista, para sorpresa de muchos "metimos" LINUX en la XBOX y dijimos que para hacerlo debíamos tener un Mod Chip instalado. Bien, pues esa cosa llamada Mod Chip es ese circuito integrado del que hemos hablado antes.

Como lo bueno no nunca viene solo, hay varios Mod Chips en el mercado, por ejemplo el que recomendamos en el número anterior, el **Aladdin Advance Xbox Modchip (ver foto)**



El Mod Chip ese que aconsejáis, ¿qué características tiene?

Mejor te pasas por aquí y lo ves tu mismo :) http://www.xboxreality.net/showdocument.php?t=270

¿Cómo se instala?

Ya dijimos que había que soldarlo, es decir, armarte de valor, abrir la consola y darle caña al soldador. Puedes encontrar mucha información sobre ello en multitud de Webs (http://www.chipeando.com/ o http://www.chipspain.com o http://www.xboxreality.net/).

¿Eres de los que no se arriesgan? Pues que te lo instalen!!!

La verdad, si nunca has cogido un soldador, mejor te instalan el chip en una tienda especializada. Como ya dijimos en el número anterior, hay quienes incluso te recogen la consola en tu casa, te instalan el chip y te la devuelven. Esto lo hacen por ejemplo en www.chipspain.com o www.artecnova.com (recordamos a los lectores que no tenemos ninguna relación con dichas tiendas).

¿Es ilegal?

NO ES ILEGAL instalar un Mod Chip, así de simple. Pero debes saber que, en el momento que abras (o te abran) la consola, perderás la garantía ofrecida por Microsoft, tu eliges!!!

¿Evolution X?

Esta es otra de las cosas que debes conocer. La XBOX tiene unas rutinas que se ejecutan cuando enciendes la consola.

Estas rutinas (software) son simples programas (para que nos entendamos). Nada mas pulsar el botón de encendido la consola accede a estas rutinas (programas) y actúa en consecuencia. El resultado visible de este proceso es el **menú que visualizamos al encender la XBOX** sin meter ningún CD.

Este **menú** es lo que llamamos "Dashboard"

Imagina que pudieses cambiar ese software, es decir, imagina que pudieses cambiar el "Dashboard".

Imagina que alguien ha programado unas rutinas nuevas y mejores, imagina que estas nuevas rutinas te permitiesen hacer todo lo que te permite el "Dashboard" de Microsoft pero además te abriese camino hacia un montón de "añadidos" (por ejemplo tener un servidor de FTP en la consola). Bien, pues eso es el Evolution X.

En este artículo te enseñaremos a sustituir el "Dashboard" Microsoft por el "Dashboard" Evolution X

Algunos también se preguntarán por qué es necesario Instalar un Mod Chip si cambiando el "Dashboard" que tiene la XBOX ya podemos tener las ventajas del Mod Chip. NO, FALSO, NO TE CONFUNDAS.

El Mod Xip te da la oportunidad de "descapar" (liberar) la consola. Gracias a ello podemos luego jugar con nuestras copias de seguridad o cambiar el "Dashboard" para conseguir algunos añadidos interesantes.

Dicho todo esto, quede claro que para cambiar el "Dashboard" debes primero instalar el Mod Chip.



Existe un bug...

Existe un Bug en la XBOX que permite modificar el "Dashboard" sin necesidad de instalar un Mod Chip :)

MANOS A LA OBRA: ¿Qué necesitamos?

Os detallo aquí una lista con lo que necesitamos para poder hacer todas estas travesuras en la consola de M$.

Para instalar Evolution X:

- Evolution X (mejor ultima versión)
- Boxplorer v 0.96 Beta o superior.
- 1 CD Regrabable o DVD.
- Consola Xbox con modchip instalado.
- Pc con grabadora de Cd´s
- Nero 6.0.0.19 o superior.

Para crear una copia de seguridad:

- Evolution X instalado.
- Cliente FTP
- Pc con tarjeta de red.
- 1 o 2 Cables RJ45.
- Xbox con modchip instalado.
- Grabadora de CD's o DVD's (depende del tamaño del juego que queramos hacer nuestro Backup).
- Nero 6.019 o superior.

Evolution X

Esto es un "Dashboard" (término que ya hemos explicado antes).

Este Dashboard tiene muchos nombres, entre ellos los más conocidos son evolution x, evox, evo-x, evo….

Puede que os estéis preguntando ¿para qué cambiar mi "Dashboard" original, si el de Microsoft es muy bonito?

Pues la respuesta es muy sencilla, este "Dashboard" viene con otras muchas utilidades, por ejemplo un Servidor FTP que usaremos para hacer copias de seguridad y manejar el HD de la consola, acceso a aplicaciones, juegos, emuladores, Sistemas Operativos, etc… que tengamos en el HD (disco duro) de la consola.

Veamos de dónde descargar las cosas necesarias:

Evox y el boxplorer los podéis encontrar en www.xbox-tribe.com y en la web de la revista (www.hackxcrack.com).

Lo mejor sería tener la última versión del evox que en estos momentos es EvolutionX Build 3935, pero en las webs que os he puesto está el EvolutionX_Build_3921, una versión anterior pero muy estable, así que trabajaremos con la versión 3921.

Para los que queráis la última versión buscad un poco en Google que seguro que lo encontráis, pero no os preocupéis ya que todos los evox se instalan y configuran de la misma forma, independientemente de su versión.

Boxplorer 0.96 Beta (o superior)

Es la herramienta que vamos a usar para copiar cosas al HD (disco duro) de la consola, en este caso el evox, pero servirá para cualquier otra cosa que queramos copiar al HD.

Tiene que ser la versión 0.96 Beta o superior, ya que en las anteriores versiones sólo se podía ver el contenido del HD y DVD, pero no se podía copiar ni manipular nada.

Cd-RW:

Como vimos en el artículo anterior, Xbox no lee los CDs normales, así que necesitaremos un CD-RW para poder instalar el evolution X.

Nero 6.0.0.19 o superior:

También lo vimos en el artículo anterior, esta versión de Nero tiene la opción necesaria para grabar los CD-RW o DVD para que la consola los pueda leer.

Tarjeta de Red y cable RJ45:

El **cable RJ45** y la **tarjeta de red** los puedes comprar en cualquier tienda de informática.

El cable RJ45 lo utilizamos para conectar el PC con la XBOX y podemos encontrarnos con dos casos, cada caso requiere un cable RJ45 distinto:

CASO 1: CABLE RJ45 CRUZADO.

Utilizaremos este cable cuando conectamos directamente el PC a la XBOX. En este caso utilizaremos un solo cable.

PC <--------------> XBOX

CASO 2: CABLE RJ45 NORMAL (también llamado NO CRUZADO).

Utilizaremos este cable cuando interponemos un elemento de red intermedio (HUB, SWITCH) entre el PC y la XBOX. En este caso necesitaremos 2 cables.

PC <----> (Router/Switch) <----> XBOX

Cliente FTP:

Todos debéis tener ya vuestro Cliente FTP favorito instalado en el PC. Yo uso el "total commander", pero vosotros podéis usar el que queráis, siempre que sepáis configurarlo.

En el número 1 de esta revista se explica perfectamente todo lo relacionado con los **Servidores FTP** y los **Clientes FTP** (puedes descargar gratuitamente el número 1 de esta revista desde **www.hackxcrack.com**).

INSTALANDO EL EVOLUTION X: PREPARANDO NUESTRAS ARMAS

Ya tenemos todo lo necesario, así que manos a la obra. Todo lo que ahora explicaremos tenemos que hacerlo desde el PC.

Primero creamos una carpeta en nuestro HD (disco duro) del PC, por ejemplo "instalacion". Dentro de esta carpeta creamos otras que se llamaran así:

C
E
F

En la carpeta E y F creamos las siguientes carpetas:

Apps → Aplicaciones
Juegos → Juegos
Emus → Emuladores
MP → Media Player

Estas carpetas (C, E, F) corresponden a las unidades del HD de la consola. Así será más fácil y menos lioso.

En esta carpeta "C", descomprimimos el evox que nos hayamos bajado. Tendremos estos archivos en la carpeta .

C:



Dejamos todo como está, exceptuando el "evox.ini" que lo editaremos con el bloc de notas (o cualquier otro editor de texto). Estaréis viendo las opciones del evolution x; ahora pasaré por cada opción explicando qué significa y qué valores acepta.

[MISC]

AutoLaunchGames = No

Si o No se auto inicia un juego al introducirlo, lo dejamos en NO.

AutoLaunchDVD = No

Si o No se auto inicia una película al introducirlo, lo dejamos en NO.

DVDPlayer ="f:\apps\dvd2.0\default.xbe"

Ruta del reproductor dvd, de momento lo dejamos como está, el mes que viene lo veremos con más detalle.

AutoLaunchAudio = No

Si o No se auto inicia un cd de música al introducirlo, lo dejamos en NO.

#AudioPlayer = "c:\xboxdash.xbe"

La ruta de reproductor por defecto al meter un cd de música, lo dejamos como está (al tener la # delante significa que está desactivado).

MSDashBoard = "c:\xboxdash.xbe"

Ruta del DashBoard original de la consola, lo dejamos como está.

UseFDrive = Yes

Usar o no la partición F, lo dejamos así de momento, por si tenemos la suerte de tener los 2 GB extra.

UseGDrive = No

Usar o no la partición G, esto sólo vale para HD mayores de 120 GB, lo veremos el mes que viene.

SkinName = Original

El nombre del tema del evox, lo dejamos como está.

IGR = Yes

Es para resetear la consola mientras estás jugando con la combinación de teclas L+R+START+BACK, pero esto crea conflictos con muchos juegos así que lo cambiamos a NO.

Lo siguiente lo dejamos como está:

#SkinName = RuDeDuDe2
UseItems = No
ScreenSaver = 5
Fahrenheit = No
ShadeLevel = 90
EnableSMART = Yes
HDD_Temp_ID = 194
DebugTSR = No
ChameleonLed = 15

[Network]

SetupNetwork = Yes

Si queremos o no una conexión a red, lo dejamos en Yes.

StaticIP = No

Queremos o no tener una ip fija. Lo cambiaremos a YES.

Ip = 192.168.0.2

Ip de la consola, que debe ser mayor que la del PC.

Subnetmask = 255.255.255.0

Mascara sub-red, lo dejamos como está.

Defaultgateway = 192.168.0.1

Puerta de enlace, aquí debe ir la ip del PC.

Lo siguiente lo dejamos como está:

DNS1 = 0.0.0.0
DNS2 = 0.0.0.0
SkipIfNoLink = No
SetupDelay = 0

[Clock]

JumpToMsDash = No
JumpIfNoLink = Yes
Use24 = Yes
SwapDate = No
SNTP_Server = 216.244.192.3

[FTP]

Enable = Yes

Si o no queremos tener acceso por FTP a la consola, lo dejamos en Yes.

Password = xbox

La contraseña del FTP, lo que queráis, yo lo he dejado como está.

IGR = No

Si o No aceptar el reseteo en caso de estar transfiriendo datos por ftp, lo dejamos en NO.

Lo demás, hasta [MENU], lo dejamos como está.

Ahora en menú tenemos que tomar algunas decisiones, como por ejemplo qué carpetas voy a tener, qué quiero que aparezca antes en el menú: los juegos que copie al HD o las aplicaciones.

Os voy a enseñar cómo modificar el menú, para que así vosotros mismos podáis hacer el vuestro. Lo que está entre comillas lo podremos traducir o cambiar, según nuestros gustos.

Este es el que viene por defecto:

[Menu]

```
Section "Root"
{
Item "Launch DVD",ID_Launch_DVD
Item "Trainers",ID_trainer
Item "MS Dashboard",ID_MS_Dash
Item "Reboot",ID_Quick_Reboot
Item "Power Off",ID_Power_Off
#Item "Flash Evox BIOS","c:\bios\d6.bin",ID_Flash_Bios_File
#Item "Lock Harddisk",@210
#Item "Unlock Harddisk",@211
Section "System Utils"
        {
Item "Settings",ID_Settings
Item "Flash BIOS",ID_Flash_Bios
Item "Backup",ID_Backup
Item "Skins",ID_Skins
        }
Section  "Launch Menu"
        {
 Section "Games"
                {
AutoAddItem "e:\games\"
AutoAddItem "f:\games\"
                SortAll
                }
Section "Apps"
                {
AutoAddItem "e:\apps\"
AutoAddItem "f:\apps\"
                SortAll
                }
        }
}
```

Y este es el mío:

```
Section "Root"
{
Section  "Menu Lanzadera"
        {
Section "Juegos"
                {
AutoAddItem "e:\juegos\"
AutoAddItem "f:\juegos\"
                SortAll
                }
Section "Aplicaciones"
                {
AutoAddItem "e:\apps\"
AutoAddItem "f:\apps\"
                SortAll
                }
Section "Emuladores"
                {
```

```
AutoAddItem "e:\emus\"
AutoAddItem "f:\emus\"
                    SortAll
                    }
Section "Media Players"
                    {
AutoAddItem "e:\mp\"
AutoAddItem "f:\mp\"
                    SortAll
                    }
          }
Item "Lanzar DVD",ID_Launch_DVD
Item "MS Dashboard",ID_MS_Dash
Item "Reiniciar",ID_Quick_Reboot
Item "Apagar",ID_Power_Off
Section "Utilidades de sistema"
          {
Item "Configuracion",ID_Settings
Item "Flashear BIOS",ID_Flash_Bios
Item "Copia De Seguridad",ID_Backup
Item "Skins",ID_Skins
Item "Formatear F:\",@210
          }

}
```

Como veréis, todo va relacionado con secciones (Section) y Funciones (Item).

Lo que hay que tener en cuenta, es que las Section pueden englobar otras Section o Item, pero los ítems sólo son ítems (funciones) y no pueden contener nada, sino que son el último "clic" donde vamos a dar para llegar a un fin, por ejemplo arrancar un juego que hayamos pasado al HD.

Las secciones empiezan por **Section "Nombre de la Section"**

Abrimos un **{** y todo lo que esté aquí será parte de la Section, y cerramos con un **}**.

La Section "Root" es la principal por lo que no podremos cambiar el nombre y todo debe estar comprendido en ésta.

Yo he puesto otra Section justo debajo de la principal, ya que es la que más uso, que tiene las opciones de:

Juegos
Aplicaciones
Emuladores
Media Players

Como veis dentro de cada apartado de la Section "juegos" hay dos AutoAddItem con dos rutas diferentes, y puede haber cuantas quieras.

AutoAddItem detecta las carpetas que hay dentro de la ruta que le hemos dado con todos los ejecutables (en Xbox son default.xbe). Con esto conseguimos que no tengamos que modificar este .ini cada vez que copiemos algo al HD de la consola, sino que lo detectará automáticamente, sólo tendremos que resetear la consola después de copiar lo que queramos al HD.

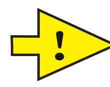
Si os gusta mi evox.ini, lo podréis bajar del foro de nuestra web, ya sabéis copiar y pegar ;-) (www.hackxcrack.com)

Bueno, ya ha acabado la configuración del evox así que ahora pasamos a descomprimir el boxplorer en la carpeta "instalacion", tal cual. El resultado de esta descompresión será una carpeta "media" y un default.xbe

Ok, ya casi estamos, ahora quemaremos (grabamos) el contenido de la carpeta "instalacion" a un CD-RW con el Nero, tal y como os enseñé el mes pasado. Si no tienes el número anterior de PC PASO A PASO, puedes pedirlos desde la Web (www.hackxcrack.com); de todas maneras, puedes encontrar el artículo de la XBOX del mes pasado LIBERADO en la misma WEB ;)

INSTALANDO EL EVOLUTION X: CONQUISTANDO NUESTRO OBJETIVO.

Todo lo que ahora explicaremos tenemos que hacerlo desde la XBOX



Debéis conocer...

Debéis conocer vuestro modchip, hay algunos modchip que detectan antes evoxdash.xbe que el xboxdash.xbe (el Dashboard original de la consola). Si vuestro chip no tiene esta función, debéis cambiar el nombre del archivo xboxdash.xbe. Por msxboxdash.xbe y evoxdash.xbe por xboxdash.xbe (también debéis cambiar la ruta de acceso en el apartado de "MS DASHBOARD", por el nuevo.)

Ahora encendemos la consola con el chip activado y metemos el CD-RW que acabamos de quemar y reiniciamos.

Nos saldrá esta pantalla:



Bienvenidos al Boxplorer, esto es un gestor del HD como pueda ser el Explorer de Windows, pero algo más simple.

Consta de dos ventanas (no se pueden ver las dos a la vez), llamadas A y B, así lo podemos ver en la esquina superior derecha.

Cuando hagamos algo, por ejemplo copiar, si pinchamos en la opción copiar en la ventana A, lo que queramos copiar pasará directamente a la ventana B, es decir, si estamos viendo el contenido de un DVD en la ventana A y en la ventana B estamos en E:\juegos\yoquese\ y pulsamos copiar el contenido seleccionado de la ventana A, irá a parar a la carpeta E:\juegos\yoquese\

Si no os ha quedado claro no os preocupéis, ahora lo veremos con un ejemplo práctico :)

La configuración del mando para el Boxplorer es la siguiente:

Cruceta digital y Stick Analógico Izquierdo: Mueve el foco.

Botón A: Entrar en una carpeta.

Botón B: Salir de una carpeta.

Botón X: Selecciona archivos.

Botón Y: Deselecciona.

Gatillo L: Cambia a la ventana A

Gatillo R: Cambia a la ventana B

Start: Seleccionar una partición.

Botón Blanco: Opciones.

Botón Negro: Salir.

Pues ahora que ya sabemos cómo funciona, y suponiendo que estamos en la ventana A, pulsamos Start y seleccionamos el DVD.

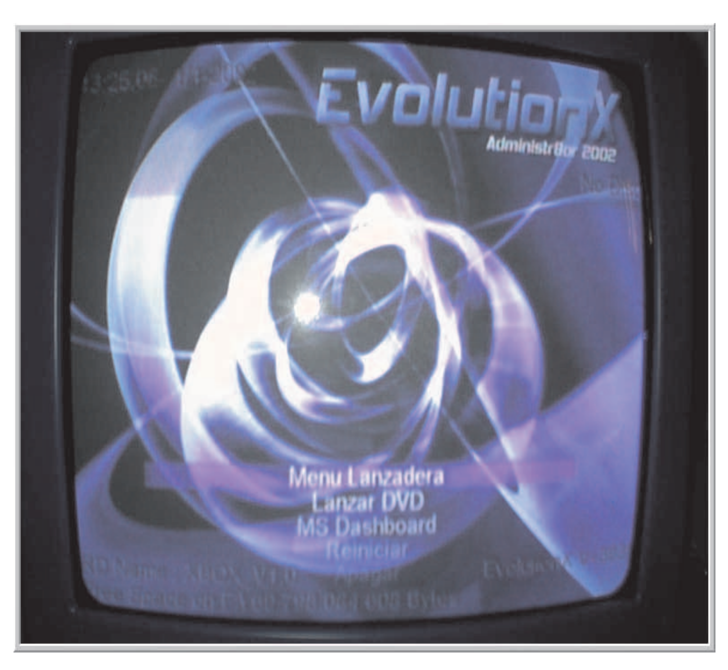
Veremos el contenido del CD-RW que tenemos metido, entramos en la carpeta C (donde está el evox), cambiamos a la ventana B pulsando el gatillo R, pulsamos Start y seleccionamos la partición C, volvemos a la ventana A y pulsamos

el botón blanco, nos saldrán las opciones, seleccionamos la que pone "MARK ALL", que nos selecciona todo, volvemos a dar al botón blanco y seleccionamos la opción "COPY", ahora nos saldrá una pantalla de aviso que nos dice que si deseamos continuar con la acción debemos pulsar los dos gatillos y el Start a la vez, lo hacemos y en nada ya está copiado a nuestra partición C.

Ahora hacemos lo mismo con el contenido de la carpeta E, pasándolo a la partición E.

Si queremos tener el Boxplorer en el HD de nuestra consola para así no tener que meter este CD cada vez que queramos copiar algo, creamos dentro de la partición E, en la carpeta "apps" una carpeta llamada "boxplorer", y aquí copiaremos la carpeta "media" y el default.xbe del CD-RW.

Sacamos el CD y reiniciamos. Si todo ha ido bien, nos saldrá esta pantalla:



Esto es el Evolution-X:
OBJETIVO CONSEGUIDO

PULIENDO DETALLES:

Si habéis usado mi evox.ini, tendremos que ir a "Utilidades de sistema".

Aquí tenemos la opción de Format F. Lo hacemos. Si tenemos suerte y nuestro HD tiene 2Gb extras, éstos ahora estarán disponibles. Para verlos, vamos a la opción de configuración y buscamos el apartado de "HARD DISCK", donde nos salen las distintas particiones con sus respectivos tamaños; si al lado de la unidad F no aparece nada, eso significa que no tenemos los 2 Gb extra.

(Para los que sí tengan la partición F: Metemos el CD en la consola y nos vamos a "Menu Lanzadera", a la opción de aplicaciones, y si habéis copiado el boxplore tal como dije, lo debéis ver. Lo arrancáis y copiamos el contenido de la carpeta "F" del CD a la partición "F").

Ahora, por ejemplo, queremos copiar algo al HD de la consola. Sólo lo tenemos que meter en el DVD y con el boxplorer pasarlo a una de las carpetas que hemos creado.

Ya está, ya tenemos configurado e instalado el EVOLUTION X, ahora lo que haremos es conectar el PC a la consola y así crear una copia de seguridad de un juego original ;)

Conectando La XBOX al PC, accediendo desde el Cliente FTP del PC al Servidor FTP de la XBOX y haciendo nuestra primera copia de seguridad.

En primer lugar instalaremos nuestro Cliente de FTP favorito en nuestro PC.

Para conectarnos al Servidor FTP de la XBOX utilizaremos la siguiente configuración: (Aquí os dejo la configuración del Cliente FTP que debéis poner si habéis usado mi .ini, si no fuese así, tendréis que sustituir estos valores por los vuestros)

Ip del Host: 192.168.0.2

Nombre de usuario: xbox

Contraseña: xbox

Modo pasivo activado



Es importante que...

Es importante que la IP del PC sea la misma que la puerta de enlace que habéis puesto en la consola. Si habéis seguido mis instrucciones las configuraciones deberán ser las siguientes:

Configuración de la Tarjeta de Red del PC:

Dirección IP: 192.168.0.1
Mascara de Subred: 255.255.255.0

Configuración del EVOLUTION X:

Dirección IP: 192.168.0.2
Mascara de Subred: 255.255.255.0
Puerta de Enlace: 192.168.0.1

Conectamos los cables JR45, encendemos la consola y metemos un juego original.

Ahora desde el PC nos conectamos al FTP de la consola como si fuera un FTP normal y corriente, si todo ha ido bien, veremos las distintas particiones del HD de la consola.

Nos metemos en la partición D (que es en realidad la unidad DVD) seleccionamos todo y lo copiamos a nuestro HD del PC.

Investigamos un poco en www.google.com si este juego en cuestión necesita algún parche (hay tantos que no puedo poner todos aquí). Si es así, lo parcheamos y luego lo grabamos con el Nero tal y como os he enseñado.



Dejamos un poco...

Dejamos un poco en el aire eso de "parchear el juego". Para quien no sepa qué es eso el mes que viene te enseñaremos a hacerlo :)

De todas maneras, por el momento, para saber si un juego necesita ser parcheado o no, pasad el juego en cuestión al HD de la consola y probad a ver si funciona (siempre y cuando el juego quepa). Si funciona, ya puedes grabarlo a un CD o DVD, si no funcionase, habría que buscar el parche.

El mes que viene detallamos el proceso ;) y veremos también cómo bajarte y grabar copias de seguridad de Internet que otras personas han hecho de sus juegos originales.

Eso es todo por este mes, espero que os haya gustado la aventura. El mes que viene, veremos cómo cambiar el HD y algunos truquitos más.

Salu2.



RAW 10

NNTP (USENET)

- PROTOCOLO NNTP (EL GRAN OLVIDADO)
- NEWS / USENET: UNA RED DE FOROS
- TOPOLOGIA DE USENET
- JERARQUIA DE NEWSGROUPS
- SERVIDORES ESCLAVOS
- LA SOCIEDAD DE USENET: TROLLS / FLAMEWARS / SPAM-BOTS / CROSSPOSTING ... ...

1. INTRODUCCIÓN

Sin duda, escribir este artículo me va a traer muchos recuerdos, ya que hacía varios años que no conectaba con un servidor de Usenet y, en cambio, cuando todavía iba en pañales por la red, "las news" ocupaban más de la mitad de mi tiempo cibernético (Usenet, news... si estos términos no te son familiares, sigue leyendo y descubrirás un nuevo mundo; si ya los conoces, profundizaremos en su funcionamiento).

Creo que puedo asegurar que Usenet fue uno de los motivos por los cuales me enamoré irremediablemente de Internet cuando, allá por el año 95, me encontré repentinamente abrumado por una cantidad de información con la cual antes ni siquiera había soñado. Por aquel entonces, mi contacto con LA RED era a través de un terminal en modo consola (texto) servido por una máquina VMS (si sientes curiosidad por este término, www.google.com). El navegar por la WWW mediante un navegador en modo texto (Lynx) hacía que la Web no fuese precisamente lo que se me presentaba como más atractivo de Internet.

Pero el hecho de que no pudiese ver las páginas con un interfaz gráfico no era el único motivo. Tan sólo unos pocos meses después instalé un prodigio de la tecnología, un módem 14400, en el flamante Pentium 90 nuevo de mi hermano. Conecté por primera vez a la red con un interfaz gráfico, el poderoso Windows 3.1, gracias al prodigioso (y ahora prehistórico) Trumpet Winsock, y pude ver al fin una página Web tal y como fue concebida, en un navegador gráfico como era Netscape (Explorer ni siquiera existía, pero seguro que tampoco lo habría usado aunque hubiese existido :-P).

Sinceramente, por aquel entonces la Web me pareció poco más que curiosa, al igual que cualquier cosa nueva que me pudiera haber encontrado, pero no me pareció realmente interesante. Al menos, no al lado de la ingente cantidad de información que circulaba por Usenet, terreno que iba conociendo ya bastante bien.

¿Cuántas horas pasaría por aquel entonces rebuscando entre los miles de grupos, suscribiéndome y desuscribiéndome y, sobre todo, fascinándome ante la enorme diversidad de culturas y personalidades con las que podía contactar un pobre chaval como yo para el que, hasta ese momento, su contacto más directo con otra cultura habían sido los comics de Marvel?

Quizá hoy día las news estén desfasadas desde el punto de vista tecnológico, pero no es fácil hacer desaparecer una sociedad formada durante años por cientos de miles, o millones de personas y, en cuestiones como ésta, donde lo realmente importante es la información, la tecnología queda en un segundo plano.

Probablemente, Usenet será conocido para muchos de vosotros, pero estoy también seguro de que más de uno no habréis oído hablar jamás de lo que sigue siendo hoy día una de las comunidades más importantes de Internet, y todo esto os estará sonando a chino.

Para todos esos que aún no sabéis de qué hablo, empezaré por una pequeña introducción para contaros qué es todo esto, porque hoy día sigue siendo una importante fuente de información que, por su carácter propio, no puede ser reemplazado por los nuevos sistemas (al menos de momento).

2. ¿QUÉ ES USENET?

Si estas leyendo esto, quizá es tu día de suerte, porque si en lugar de habérmelo "preguntado" a mí, hubieses buscado respuesta a esta pregunta en los FAQs y diversos artículos acerca de Usenet te habrías encontrado con multitud de respuestas evasivas y complicadísimas reflexiones filosóficas y metafísicas acerca de lo que es, no es, deja de ser, y sigue siendo Usenet. Personalmente, no entiendo bien a qué se deben estas comeduras de cabeza porque, si bien Usenet es complejo en sus detalles, la idea general es bien sencilla. Esto es lo que es Usenet: **una red de foros**.

Supongo que todos sabéis lo que es un foro: un punto de encuentro para gente interesada en un tema común, en el cual esta gente puede escribir artículos, preguntas, o cualquier otra cuestión que deseen compartir o comunicar al resto de personas del foro (claro ejemplo es el foro de esta revista: www.hackxcrack.com). Usenet es básicamente una comunidad de foros en la cual existen decenas de miles de foros diferentes, cada uno con su temática particular y sus usuarios particulares, y donde se pueden crear nuevos foros según vaya surgiendo la necesidad por parte de los usuarios.

Quizá algunos conozcáis Usenet por otros nombres más comunes, ya que vulgarmente se le suele llamar "las news" o "los newsgroups".

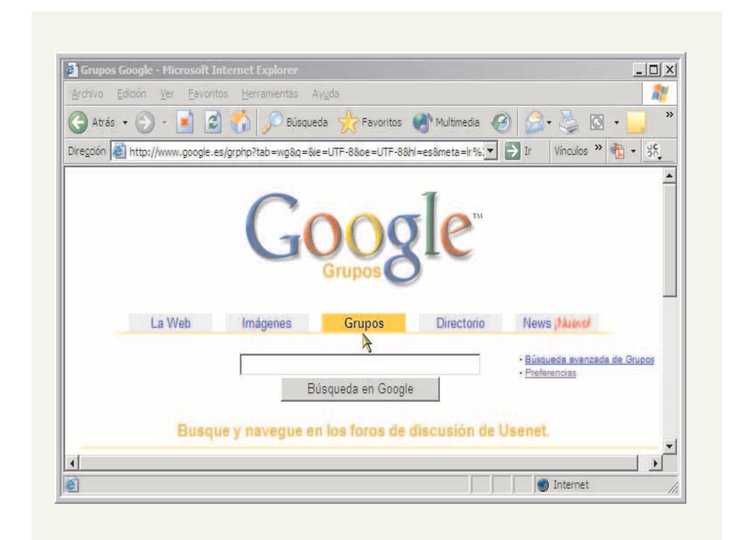
Posiblemente más de uno os habréis topado con estos términos pero los habréis ignorado al no saber de qué iban. Por ejemplo, al contratar vuestra conexión a Internet, posiblemente vuestro ISP os dio la dirección de su "servidor de news", o al configurar vuestro cliente de correo (Outlook, Netscape, ...) os habréis encontrado con opciones para conectar con un "servidor de news", o incluso en el propio Google habréis visto el apartado GRUPOS entre los 5 apartados principales: La web, Imágenes, Grupos, Directorio, y News (en este caso, News no tiene nada que ver con el tema, pero si Grupos).



Hay un dicho poular...

Hay un "dicho" popular que reza: "si no sale en la –tele-, no existe", pues hoy en día podríamos decir lo mismo pero relacionado con Internet: "si no sale en GOOGLE, no existe".

Hay muchas utilidades del buscador GOOGLE que muchas personas desconocen. Una es la de **poder buscar imágenes** (botón **imágenes**) y otra es la de poder buscar en USENET (botón **grupos**). Dedícale 3 minutos a cada opción y **ya no podrás vivir sin ellas**.



No es casualidad que la mayoría de clientes de correo incorporen también funciones para conectar con Usenet, ya que muchos de los conceptos que manejaremos serán muy parecidos a los de una mailing-list, o lista de correo, y el formato de los mensajes en Usenet es bastante similar al del correo electrónico.

2.1. Jerarquía de los newsgroups

Antes de seguir por el camino que he empezado a tomar, voy a avisaros de que hay gente bastante quisquillosa a la cual no le gusta que se hable de "foros" cuando se habla de Usenet, así que a partir de ahora usaremos el término **grupo** para referirnos a un foro concreto de Usenet.

Quizá os preguntéis de qué forma se pueden administrar tantísimos grupos (he hablado antes de decenas de miles). Realmente es muy difícil mantener el orden en tal cantidad de información, ya que son miles de artículos nuevos los que aparecen cada día publicados en los distintos grupos de Usenet, y en la práctica, no sólo es difícil, si no que de hecho no se ha conseguido.

Si buscáis en Usenet un sistema fiable de comunicación, estáis totalmente equivocados y ya os podéis ir buscando otra cosa. Más adelante os hablaré de los problemas que puede haber con los artículos de un grupo, pero de momento vamos a tratar de los problemas que hay con la propia existencia de los grupos.

Para organizar la ingente cantidad de información, en Usenet existe una ordenación jerárquica. En los comienzos de Usenet (a principios de los años 80, aunque hasta el año 86 no se impuso el protocolo **NNTP** sobre el **UUCP**, utilizado anteriormente) se crearon 8 grupos principales de los cuales colgarían todos los demás mediante una estructura en árbol como la que siguen los directorios (carpetas) de un disco duro. Estos 8 grupos principales eran los siguientes:

comp.* - (computers) todo lo relacionado con ordenadores, informática, etc.
humanities.* - como bien dice el nombre: humanidades.
news.* - todo lo relacionado con la propia Usenet.
rec.* - (recreation) ocio.
sci.* - (scientific) ciencia.
soc.* - sociedad.
talk.* - grupos para charla.
misc.* - cualquier tema que no encajase en alguna de las otras categorías.

Cada una de estas categorías principales fue ampliándose con multitud de subcategorías, pero también se fueron creando con el tiempo nuevas categorías principales, habiendo cientos de ellas actualmente. La más importante de todas las categorías nuevas fue quizá la categoría **alt.***, que englobaba todos los grupos ALTernativos no creados dentro de las categorías principales (y de la cual actualmente cuelgan más de 20 mil grupos), aunque también cabe destacar la categoría **biz.***, dedicada a negocios, y otras categorías que abarcan varios miles de grupos, como **free.*** o **microsoft.***. Además, se crearon diversas categorías nacionales: **es.*** (España), **fr.*** (Francia), etc.

Si entramos en el apartado GRUPOS de www.google.com veremos un directorio bastante extenso de los grupos de Usenet que existen en la actualidad. En la página principal sólo se muestran las categorías principales, pero puedes expandir la lista completa de grupos para hacerte una idea de la cantidad de grupos que existen.

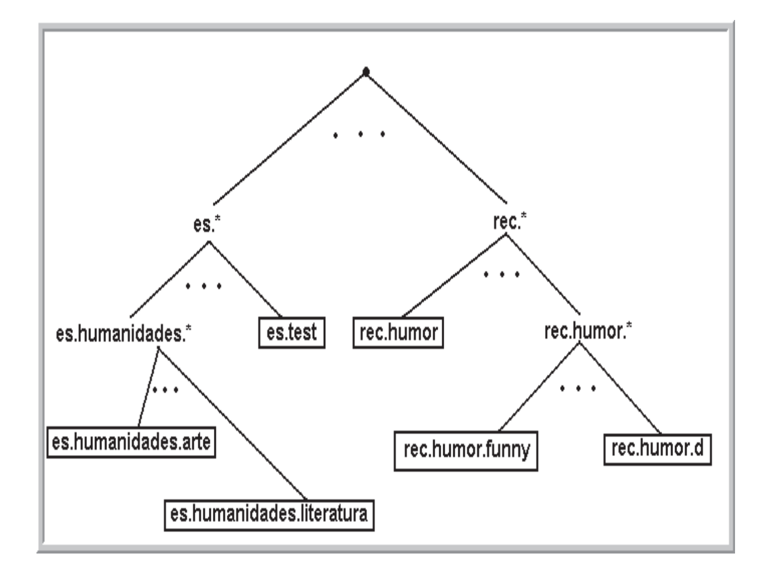


Fig 1.- Jerarquía de grupos de Usenet.

En esta ilustración distingo las categorías de los grupos recuadrando a estos últimos. Como vemos en el ejemplo, de la categoría **es.*** cuelgan muchos grupos, como por ejemplo el grupo **es.test** (grupo español para realizar pruebas), pero también muchas subcategorías, como por ejemplo la categoría **es.humanidades.***. A su vez, de la categoría **es.humanidades.*** cuelgan, por ejemplo, los grupos **es.humanidades.arte** y **es.humanidades.literatura**.

Entre los muchos grupos y las muchas categorías que cuelgan de la categoría principal **rec.***, encontramos como ejemplo curioso el del grupo **rec.humor** que, además de ser un grupo, existe también una subcategoría (**rec.humor.***) con el mismo nombre, de la cual cuelgan por ejemplo los grupos **rec.humor.funny** y **rec.humor.d** (vete tú a saber qué será este último, porque el nombre no es muy explícito que digamos...).

Para acceder a Usenet basta con conectarse a uno de los miles de servidores existentes mediante el software adecuado (clientes de correo con funciones para lectura de news, o bien clientes específicos), que lo que hace internamente es comunicarse con el servidor de news, a través del puerto **TCP 119**, mediante el protocolo **NNTP**, que es precisamente el que detallaré en este artículo. Una vez conectados a la red, solicitaremos al servidor una lista de los grupos disponibles, y navegaremos por la jerarquía para **suscribirnos** a aquellos que nos interesen.

Desde el momento en que nos suscribimos a un grupo, ya podremos consultar los artículos específicos publicados en ese grupo y, si el grupo está configurado para permitirlo, también enviar nuestros propios artículos para que los puedan ver el resto de usuarios suscritos a ese grupo.

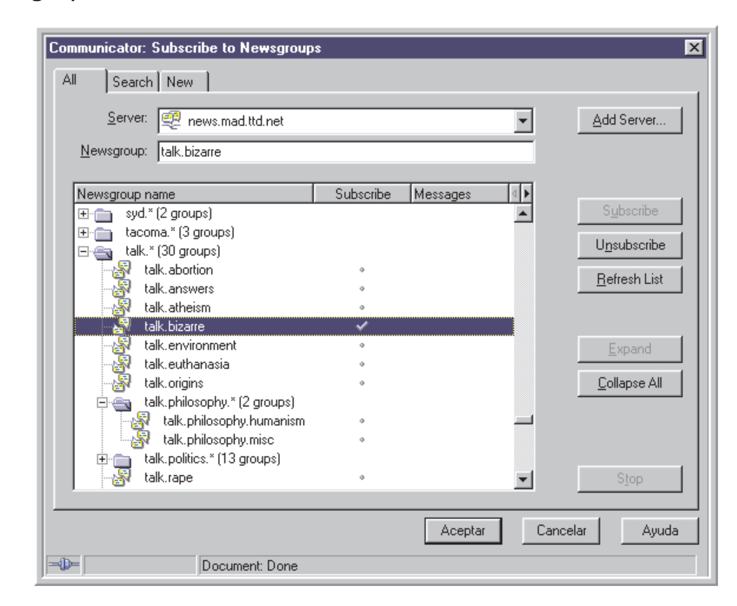


Fig 2.- Listado de grupos del servidor news.mad.ttd.net, utilizando como software Netscape Messenger, en la cual nos acabamos de suscribir al grupo talk.bizarre, para hablar de cosas extrañas...



Hay un dicho poular...

Configurando nuestro software para leer las NEWS y buscando un Servidor de Noticias gratuito.

Como no podemos negar la realidad, seguro que la mayoría de lectores utilizan WINDOWS XP como sistema operativo. Todo usuario del Sistema Operativo Windows tiene instalado el programa Outlook Express (es parte del sistema operativo) que sirve para enviar y recibir mails (gestor de correo electrónico); pero muchos no saben que ese mismo programa se utiliza también para acceder a USENET y leer “las news” (gestor de grupos de noticias).

Antes de seguir, un apunte importante: Si alguien tiene instalado el OFFICE y utiliza como gestor de correo electrónico el Microsoft Outlook (incluido en el OFFICE), que se olvide de acceder a “las news”. El Microsoft Outlook no tiene gestor de news!!!

Por lo tanto debemos iniciar el Outlook Express (Menu Inicio --> Todos los Programas --> Outlook Express) y una vez abierto el programa crearemos una “cuenta de noticias”. Para crearla iremos al Menu Herramientas --> Cuentas

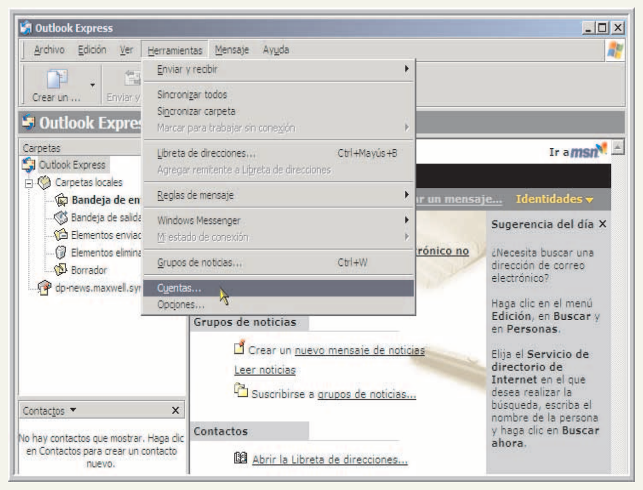


Imagen A

Aparecerá una ventana y del grupo de botones que hay a la derecha pulsaremos sobre AGREGAR NOTICIAS.

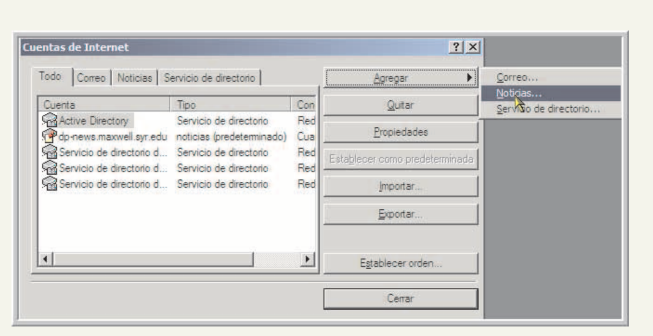


Imagen B

Ahora aparecerá una ventana donde nos preguntará el nombre, podemos escribir lo que queramos, tu nombre real (por ejemplo PEDRO) o un NICK (por ejemplo BIGBOY). Nosotros pondremos MAXLINE.

Pulsaremos el botón siguiente y nos pedirá una Cuenta de Correo electrónico, podemos poner lo que queramos, por ejemplo una que no exista, noexisto@losiento.com (es una buena practica para evitar el correo basura). Pulsaremos el botón siguiente y nos aparecerá la tercera y última ventana del asistente, donde se nos pide el Servidor de Noticias.

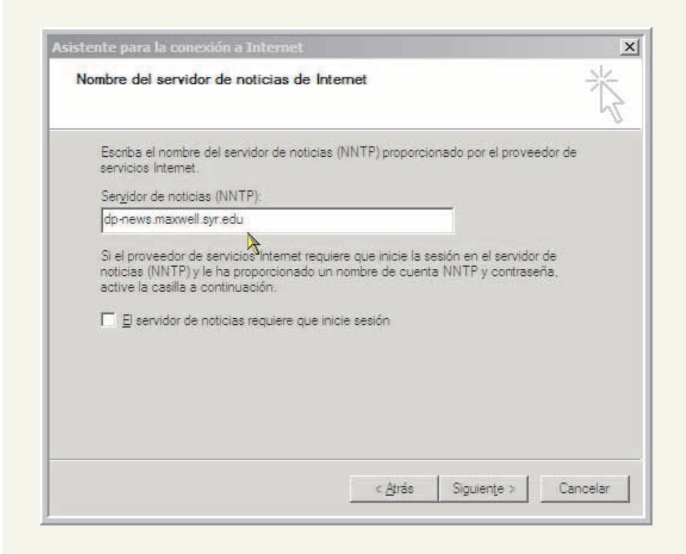


Imagen C

Esta es la ventana más importante y deberemos poner el Servidor de News que nos ofrece nuestro ISP (nuestro Proveedor de Internet, por ejemplo Telefónica); si no dispones de ese dato, llama a tu ISP y exige que te lo proporcionen.

Aunque es recomendable utilizar el Servidor de News de tu ISP, quizás te interese probar con servidores Gratuitos (o Libres, según se mire). Hoy en día es difícil encontrar un Servidor de Noticias Gratuito y que permita "a cualquiera" conectarse a ellos, nosotros utilizaremos el Servidor de News **dp-news.maxwell.syr.edu** (que figura en la imagen anterior y en la fecha de publicación de esta revista funciona perfectamente, aunque no te deja). Puedes probar con cualquier otro de la lista http://www.newzbot.com.

Una vez introducido pulsaremos el botón siguiente y en la siguiente ventana pulsaremos el botón finalizar. Nos encontraremos entonces frente a la ventana de la **Imagen B**, donde pulsaremos sobre el botón CERRAR. Después nos preguntará si queremos descargar los Grupos de Noticias a lo que responderemos que si. Si todo ha ido bien, se descargarán los grupos de noticias y podrás suscribirte a los que quieras.

Aquí te dejamos para que experimentes tu mismo o corremos el riesgo de que en el foro nos machaquen a críticas por explicar cosas tan sencillas. Si tienes dudas sobre el funcionamiento del Outlook Express pregunta en el foro de la revista (www.hackxcrack.com).

2.2. Topología de la red Usenet.

Pero el fondo de la cuestión es ahora: ¿Cómo se administra toda esta cantidad de grupos? ¿Cómo se consigue que en todos los servidores de Usenet existan los mismos grupos con los mismos artículos? Pues la respuesta es bien sencilla: NO se consigue. :-)

En el artículo sobre DNS vimos cómo se administraba este sistema, también jerárquico, centralizando cada subdominio en una serie de servidores dedicados a ello. En cambio, en Usenet no hay ningún tipo de centralización, así que todos los servidores pueden tener a todos los grupos.

En la práctica, no hay ningún servidor que tenga todos los grupos, ya que cada servidor está configurado de una forma diferente, según los intereses de cada administrador (intereses de una compañía, si es por ejemplo un servidor perteneciente a un ISP, o bien intereses personales, si es por ejemplo un servidor perteneciente a un usuario particular como tú o como yo que ha decidido ofrecer altruistamente su propio PC como servidor para toda la comunidad de Usenet).

Según esta configuración, cada servidor incluirá en su lista unos grupos u otros. Por ejemplo, puede haber servidores censurados que no incluyan en su lista ningún grupo relacionado con sexo o violencia, o puede haber servidores españoles que sólo incluyan grupos internacionales o españoles, pero no grupos franceses, japoneses, o de otros países, etc., etc.

El problema es que este tipo de filtrado de grupos puede ser propagado a otros servidores en contra de sus deseos. Por ejemplo, veamos la siguiente ilustración:

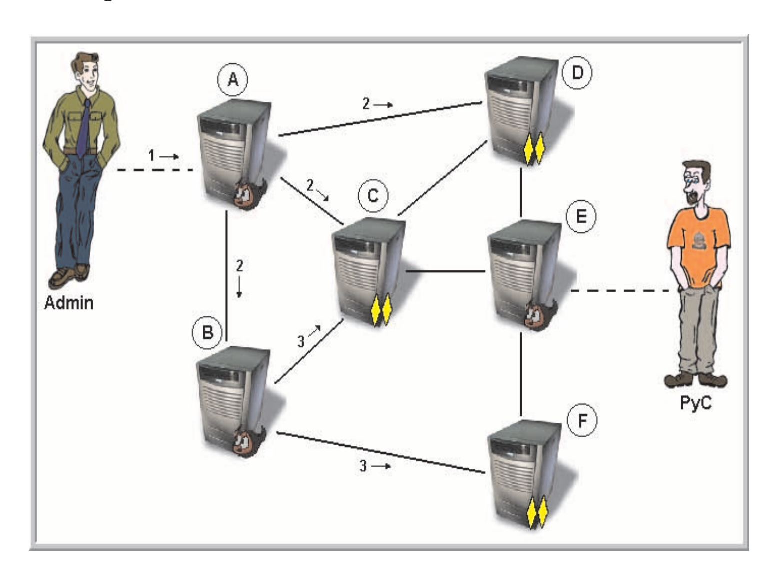


Fig 3.- Topología simplificada de la red Usenet.

En el ejemplo vemos una red simplificada que consta tan sólo de 6 servidores, aunque en

realidad Usenet consta de miles de servidores. Como ejemplo, podéis encontrar listas de servidores en http://www.newzbot.com/.

En nuestra red ficticia existen dos tipos de servidores: los servidores malignos, y los servidores católicos. Los servidores **malignos** (señalados con un pequeño diablillo en la ilustración) no tienen ningún tipo de censura, mientras que los servidores **católicos** (señalados con unos rombos en la ilustración) aborrecen los contenidos pornográficos.

Supongamos que el administrador (**Admin**) del servidor **A**, que es uno de los servidores **malignos**, decide crear un grupo nuevo, llamado **alt.sex.aliens**, para los usuarios que desean tratar el apasionante tema de las relaciones sexuales con seres de otros planetas (aunque parezca mentira, este grupo existe en realidad).

En un primer instante, el administrador creará el grupo en la lista del servidor **A**.

A continuación, en el instante 2, el nuevo grupo se propagará a todos los servidores conectados al servidor **A**, es decir: los servidores **B**, **C**, y **D**.

El servidor **B**, uno de los **malignos**, incluye el nuevo grupo en su lista, ya que no tiene ningún filtro programado para no incluir ese tipo de grupos. En cambio, tanto el servidor **C** como el **D** pertenecen a los servidores **católicos**, y tienen programado un filtro que impide la creación de ningún grupo que contenga palabras como "sex".

En el tercer instante, el servidor **B**, el único que incluyó el nuevo grupo en su lista, intentará propagarlo hacia todos los servidores conectados a él. Por supuesto, no intentará propagarlo hacia el servidor **A**, a pesar de que esté conectado a él, ya que sabe que el servidor **A** ya conoce el grupo, pues fue él mismo el que le comunicó que el grupo había sido creado (el cómo sabe exactamente el servidor **B** que el servidor **A** ya conoce ese grupo es algo que explicaré más adelante). A los que si que intentará propagar el nuevo grupo es a los servidores **C** y **F**.

El servidor **C** pasará del tema, porque ya conocía la existencia del nuevo grupo gracias al servidor **A**. En cambio, el servidor **F** no tenía noticia de la creación del nuevo grupo, pero tampoco lo incluirá en su lista, ya que es uno de los servidores católicos, y su filtro encontrará la temida palabra "sex" en el nombre del grupo. Por tanto, los servidores **C**, **D**, y **F** han recibido la noticia de que ha sido creado el grupo **alt.sex.aliens**, pero ninguno de ellos lo ha añadido a sus listas y, por los mismos motivos, tampoco contribuirán a la propagación de ese nuevo grupo.

El pobrecito servidor **E**, que ha tenido la mala suerte de estar conectado sólo a servidores católicos (**C**, **D**, y **F**) jamás se enterará de que el nuevo grupo fue creado y, aunque a él le habría interesado añadirlo a su lista, no podrá hacerlo.

Así, cuando el pervertido de **PyC** se conecte a su servidor E para contactar con alguna guapa alienígena, encontrará sus intenciones frustradas al no existir ningún grupo adecuado para el tema.

Quizá ahora empezaremos a comprender por qué cuando se buscan definiciones de Usenet se encuentran multitud de respuestas vagas, y de cuestiones casi metafísicas. Si este nuevo grupo existe sólo en unos servidores y no en todos, ¿se considera que ese grupo pertenece a Usenet? ¿Dónde están los límites? ¿Usenet abarca todos los grupos que existen en todos los servidores, sólo aquellos que están en todos, sólo aquellos que existen en la mayoría...?

Pero éste no es el único problema, ya que lo mismo que ocurre con la creación de grupos ocurre también incluso con los artículos que se publican dentro de cada grupo. Cada servidor puede tener sus propias reglas de filtrado para censurar los contenidos que desee, o para evitar el spam, o para evitar la saturación del servidor (prohibiendo el envío de archivos binarios, que ocupan mucho más que el texto), o por cualquier otro motivo.

Al final, lo que resulta de todo esto es que Usenet no es un concepto absoluto, si no relativo, y es diferente según el servidor que usemos como punto de referencia.

Pero aún podemos rizar más el rizo, e incluso decir que la única diferencia entre un servidor y otro no es el número de grupos y de artículos que hay en cada uno, ¡si no incluso el orden en que aparecen los artículos! Muchas veces nos encontraremos con paradojas, como respuestas a artículos que llegan antes que el propio artículo original, o incluso respuestas a artículos inexistentes. Todo esto se debe al mecanismo de distribución de los artículos, que no es ni mucho menos óptimo.

Además, para colmo, los artículos tienen una caducidad dependiendo de los caprichos de cada servidor. Por ejemplo, en un servidor español podrían seguir la política de mantener los artículos de los grupos franceses sólo durante un día, y los españoles conservarlos por una semana. En cambio, en otro servidor que fuese francés, la política podría ser justo la contraria.

Siguiendo con los aspectos de la arquitectura, es importante también hablar de los **servidores esclavos**. Un servidor NNTP esclavo no realiza todas las funciones que realiza un servidor normal, si no que actúa sólo como intermediario entre la red Usenet y algún grupo concreto de usuarios, por ejemplo los usuarios de una LAN (red de área local), o los clientes de un determinado ISP.

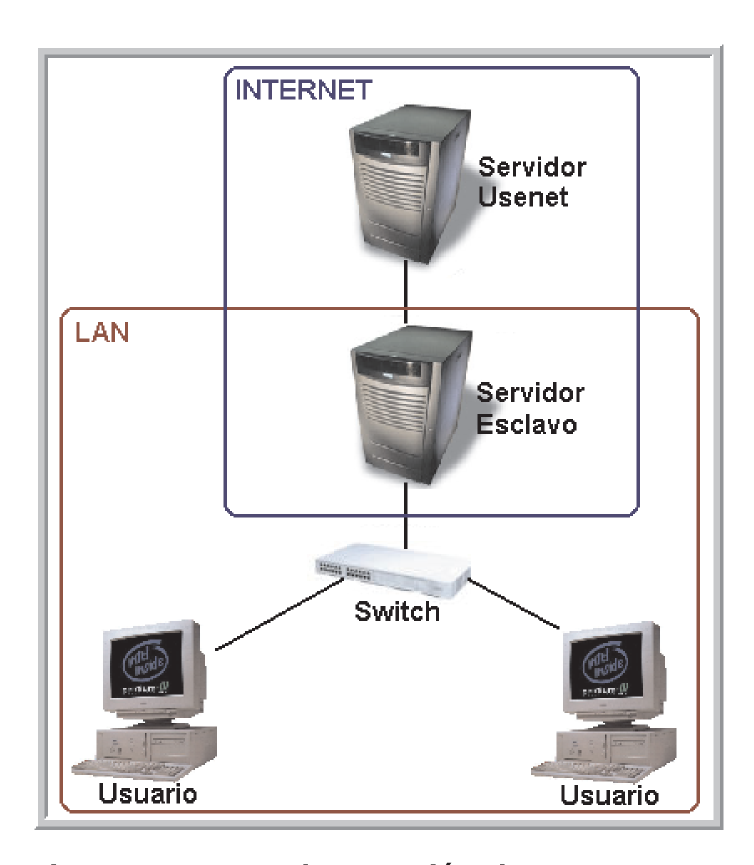


Fig 4.- Esquema de conexión de un servidor NNTP esclavo con la red Usenet.

El servidor esclavo poseerá una amplia cache para minimizar el número de accesos a la red Usenet, por lo que el uso de servidores esclavos para los distintos ISPs, LANs, o cualquier otra comunidad, puede aligerar bastante la carga de la red. Por este motivo, el **RFC** del protocolo **NNTP** recomienda que se de cierta prioridad a los servidores esclavos frente a los usuarios normales. Ya veremos más adelante algo más acerca de esto.

2.3. La sociedad de Usenet

Antes de entrar de lleno con el tema principal del artículo (el protocolo NNTP), os comentaré algunos aspectos curiosos sobre este mundillo.

En cualquier FAQ sobre el tema tendréis explicaciones y divagaciones más detalladas sobre cualquier aspecto particular del mundo

de Usenet, así que sólo os introduciré un poco el tema. Básicamente lo que haré será advertiros de algunas cuestiones que irritan bastante a los usuarios "honrados" de Usenet, para evitaros problemas si pensáis incorporaros a esta interesante comunidad.

En primer lugar, comenté algo antes acerca del **spam**. Como ya sabemos, el spam es esa odiosa publicidad que no solicitamos pero que nos llega una y otra vez impidiéndonos centrar nuestra atención en lo que realmente nos interesa. Usenet no se libra del spam, ni mucho menos, ya que existen bots que lo que hacen simplemente es conectarse a los servidores NNTP para enviar automáticamente una serie de mensajes a varios grupos.

La solución más sencilla para estos bots spameadores sería enviar un único mensaje, e indicarle al servidor que publicase ese mismo mensaje en tropecientos grupos que él especificase. El problema que tienen los bots con esto es que, en general, los servidores NNTP no se lo pondrán tan fácil.

En efecto, se puede indicar a un servidor que un mismo artículo sea publicado en más de un grupo simultáneamente, pero en general los servidores NNTP limitarán mucho esta característica, o incluso directamente la inhabilitarán.

Al limitar la publicación en múltiples grupos, no sólo están combatiendo el spam, si no que también están evitando otro problema común en Usenet, que es el conocido como **crossposting** (publicación cruzada). Cuando un usuario empieza un tema de discusión enviando un mismo artículo a varios canales, y la gente se pica y empieza a publicar respuestas al tema se puede montar un buen jaleo, pues habrá multitud de mensajes cruzados entre varios grupos simultáneamente, a los cuales será muy difícil seguirles el hilo. Ahora que menciono la palabra **hilo**, os comento como curiosidad que una traducción al inglés sería la palabra **thread**, que seguro que conocéis todos aquellos que utilicéis habitualmente cualquier tipo de foro, pues un thread es precisamente eso: un hilo de conversación.:-)

Por tanto, os aconsejo que evitéis en la medida de lo posible enviar un mismo mensaje a varios grupos, ya que podéis causar problemas y, sin duda, molestar al resto de usuarios.

En lo que respecta al spam, si el servidor limita este tipo de mensajes dirigidos a varios grupos simultáneamente, los bots pueden saltarse esto sencillamente por fuerza bruta, es decir, enviando el mismo mensaje a cada grupo por separado. En este caso, es bastante más complicado evitarlo, y la principal arma son ahora los filtros que analicen los mensajes tratando de encontrar algo sospechoso (remitentes indeseados, palabras clave como "enlarge your penis", o cualquier otro tipo de filtrado inteligente).

Ya que he hablado de las conversaciones cruzadas difíciles de seguir a las que se llega debido al crossposting, también hablaré de otras conversaciones indeseables, que son las famosas **flamewars**. Si tratáis de documentaros acerca de Usenet por vuestra cuenta, probablemente daréis con términos como flamewar, o troll. Por cómo se utilizan estos términos en la documentación que se suele encontrar, quizá podáis pensar que se trata de términos técnicos probablemente relacionados con alguna compleja técnica utilizada por "hackers" avanzados. Nada más lejos de la realidad, ya que una flamewar no es más que un hilo de conversación molesto y que no lleva a ninguna parte, ya que consiste simplemente en un intercambio de insultos, o en un diálogo de besugos. Se habla tanto acerca de las flamewars por el simple hecho de que son bastante frecuentes y engorronan considerablemente los hilos realmente interesantes de los grupos.

Para aclarar el otro término, un **troll** es simplemente lo que en cristiano podemos llamar un “tocanarices”, es decir, un tío que no tiene nada mejor que hacer que pasar su tiempo insultando a los usuarios de un grupo, o planteando cuestiones irrelevantes sólo por el gusto de crear polémica. En este punto, no todo el mundo está de acuerdo con la terminología, ya que algunos llaman troll a la persona, mientras que otros llaman a la persona **troller**, y troll a los mensajes indeseados que éstos publican en algún grupo. Si encontráis en un grupo un mensaje del tipo “sois todos unos lamers”, sin duda habéis dado con un troll.

Por último, antes de que os pongáis a hacer vuestras propias pruebas, os comento algunos puntos más. En primer lugar, si queréis probar la creación de nuevos grupos, mejor que desistáis en el intento. Crear un grupo no es sencillo, y además necesitáis el apoyo de otros usuarios, por lo que os aconsejo que primero os hagáis bien con el funcionamiento de la red, y luego os planteéis de nuevo si realmente os interesa crear un grupo nuevo (con la cantidad de grupos que existen actualmente, parece realmente difícil no encontrar al menos uno que encaje en el tema que deseáis tratar).

Con respecto a las pruebas que queráis hacer para publicar artículos, ¡no lo hagáis en cualquier grupo! Existen grupos específicos para pruebas, como por ejemplo: **alt.test**, **misc.test**, o **alt.binaries.test** (este último para pruebas con archivos binarios).

Ya que hablo de los archivos binarios, es decir, aquellos que no son texto puro (imágenes, sonidos, programas, etc), os comento que el tema tiene bastante miga. No se pueden enviar archivos binarios a cualquier grupo, si no sólo a aquellos específicamente creados para ello (generalmente, se encuentran en la categoría **alt.binaries.***). Muchos servidores directamente no incluirán ninguno de estos grupos, ya que aumentan considerablemente la cantidad de recursos necesarios (espacio de disco, y ancho de banda, principalmente). Usenet es principalmente una red de intercambio de artículos, entendidos estos como texto puro y duro. Por supuesto, siempre podéis utilizar URLs para llevar al resto de usuarios a cualquier otro tipo de contenidos (páginas web, ftp, etc).

Con respecto también al envío de artículos, algunos grupos están moderados, es decir, cualquier artículo que se desee enviar pasará previamente por un proceso de selección para juzgar (bien por una persona, o bien por un script) si conviene su publicación.

3. USENET EN PROFUNDIDAD

Vamos a ver ahora en detalle cómo funciona exactamente la red Usenet. Empezaré comentando el sistema de distribución de artículos entre los servidores, para luego pasar a detallar el protocolo NNTP y, por último, hablar acerca del formato de los artículos.

3.1. Jugando a los cromos

El sistema de distribución de artículos y grupos de Usenet es bastante parecido al juego de los cromos al que todos hemos jugado de pequeños. Primero uno mostraba sus cromos al otro, el cual iba diciendo “sile” o “nole”, y luego el otro hacía lo mismo con los suyos.

Vamos a ver la interacción entre dos servidores NNTP, uno de los cuales será el que originalmente querrá cambiar cromos (**receptor**), y el otro el que los ofrece (**emisor**).

Estos son los pasos que se seguirán normalmente:

a) – El receptor pregunta al emisor si ha incluido en su lista grupos nuevos.

b) – El emisor responde, dando la lista de grupos nuevos.
c) – El receptor, según sus propios criterios, decidirá si incluir o no los grupos en su propia lista.
d) – El receptor pregunta al emisor si existen nuevos artículos en los grupos que comparten.
e) – El emisor responde con una lista de artículos nuevos
f) – El receptor solicitará al emisor aquellos artículos que no tenga de la lista que le facilitó en el paso anterior.
g) – El emisor transmite cada artículo solicitado por el receptor. Los pasos f y g se repiten para cada artículo que pueda interesar al receptor.
h) – El receptor le facilita al emisor la lista de artículos nuevos que tiene él, por si alguno le interesa al emisor.
i) – El emisor solicita al receptor aquellos artículos que no tenga de la lista que le facilitó el receptor en el paso anterior.
j) – El receptor transmite cada artículo solicitado por el emisor. Los pasos i y j se repiten para cada artículo que pueda interesar al emisor.

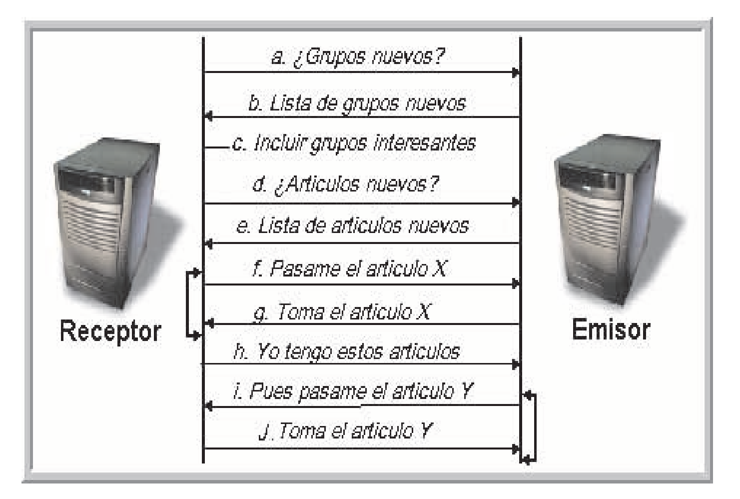


Fig 5.- Esquema del sistema de distribución de grupos y artículos entre dos servidores NNTP.

Este es el sistema que aconseja utilizar el **RFC 977**, que trata sobre el protocolo **NNTP**. En la práctica, es más eficiente propagar los nuevos grupos y artículos según se van creando, en lugar de esperar a que el resto de servidores nos lo pidan, ya que todo funciona más rápido mientras los contenidos están en la cache del servidor, y no arrinconados en su disco duro. Esto se hará mediante unos mensajes de control especiales que veremos más adelante.

Hay que destacar el hecho de que no existen comandos específicos de autenticación ni de identificación de ningún tipo para que un servidor distinga cuando se conecta un usuario de cuando se conecta otro servidor, excepto para el caso de servidores esclavos (tal y como veremos más adelante). Por tanto, la condición de usuario o de servidor se deduce implícitamente según los comandos que se ejecuten a lo largo de la sesión (un usuario suele utilizar los comandos para leer y publicar artículos, mientras que un servidor suele utilizar los comandos de distribución de artículos).

Por otra parte, cualquier servidor NNTP debería conocer las IPs de los demás servidores NNTP que se puedan conectar a él, para así no dar privilegios de servidor a un simple usuario que pudiera tener malas intenciones.

3.2. El protocolo NNTP

Al fin nos centramos ya en el propio protocolo RAW que, tal y como os acabo de comentar, se encuentra especificado en el **RFC 977** (ftp://ftp.rfc-editor.org/in-notes/rfc977.txt). Iré mostrando una sesión de ejemplo paso a paso, tal y como he hecho otras veces.

3.2.1. Conectando con el servidor.

¿Tenéis ya preparados esos **Telnets**? ¡Pues vamos allá!

telnet news.mad.ttd.net 119

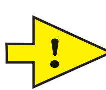
Si sois chicos listos, habréis deducido que el puerto asignado al servicio **NNTP** es el **119**. ;-)

Recuerda que en color verde tenemos **las ordenes de conexión** (ejecución del programa Telnet y conexión a un Servidor Remoto), en color rojo tenemos las respuestas del servidor y en azul los comandos que nosotros le enviamos al Servidor Remoto una vez nos hemos conectado mediante Telnet).



IMPORTANTE:

IMPORTANTE: Servidores de News que no funcionan. Usaremos este servidor (**news.mad.ttd.net**) para el ejemplo. Si no te funcionase correctamente utiliza el servidor gratuito mencionado al principio de este artículo (**dp-news.maxwell.syr.edu**) y, por si acaso, tienes listas de servidores en http://www.newzbot.com (sírvete tu mismo ;).



¿No sabes que ...

¿No sabes qué es TELNET? Malo, malo… eso significa que no has seguido nuestra revista :)

Hemos liberado muchos artículos en nuestra Web (www.hackxcrack.com), si quieres empezar a trabajar con Telnet nada mejor que descargarte el Artículo RAW 1 / POP 3 del número 7 de PC PASO A PASO y pegarle un vistazo.

Que lo disfrutes!!!

Al igual que en otros muchos protocolos, todas las **respuestas del servidor** comienzan siempre por un código numérico de respuesta de 3 dígitos.

Existen 5 tipos básicos de respuestas:

- **1xx** – Las respuestas cuyo código empieza por 1 son de carácter **informativo**.
- **2xx** – Las que empiezan por 2 indican que el comando se ha realizado con **éxito**.
- **3xx** – Las que empiezan por 3 indican que el comando es correcto y que podemos **continuar** el proceso.
- **4xx** – Las que empiezan por 4 indican que el comando es correcto, pero que **no se puede ejecutar** por algún motivo.
- **5xx** – Las que empiezan por 5 indican que el comando es **incorrecto**, no está implementado, o que hay algún problema en el servidor.

Dentro de cada uno de los 5 tipos básicos de respuestas, existe a su vez otra clasificación, en función del segundo dígito:

- **x0x** – Se refiere a cuestiones acerca de la propia **conexión** o de la **configuración** en general.
- **x1x** – Se refiere a cuestiones sobre los **grupos.**
- **x2x** – Se refiere a cuestiones acerca de los **artículos.**
- **x3x** – Se refiere al mecanismo de **distribución** de artículos y grupos.
- **x4x** – Se refiere a la **publicación** de artículos.
- **x8x** – Se refiere a cualquier función propia de un software específico, **que no forme parte del estándar**.
- **x9x** – Reservado para funciones de **depuración**.

Así, por ejemplo, una respuesta del tipo **11x** nos informará acerca del estado de nuestra conexión, una respuesta del tipo **24x** nos indicará que la publicación de un artículo ha tenido éxito, y una respuesta del tipo **58x** nos indicará que el software que corre en el servidor no tiene implementado el comando no estándar que se ha solicitado.

Todo esto lo cuento ahora porque, nada más conectar, el servidor nos enviará un código de respuesta que típicamente (si todo va bien) será uno de estos dos:

-200 – Servidor preparado.

- **201** – Servidor preparado, pero que sepas que no puedes publicar artículos a través de este servidor.

En general, para nuestras pruebas nos servirá cualquiera de los dos tipos, pero cuando lleguemos al punto de experimentar con las funciones de **publicación** de artículos (por supuesto, si son pruebas, tendremos que hacerlas en algún grupo destinado a ello, como por ejemplo **alt.test**), necesitaremos un servidor que nos responda con código **200** al conectar.

3.2.2. ¡Ayuda, por favor!

Antes de hacer nada podemos pedir ayuda al servidor, para que nos informe de los comandos que tiene implementados, así como de su formato:

HELP

El servidor normalmente nos responderá con una lista de comandos con sus parámetros, y posiblemente referencias a información complementaria o a alguna dirección para avisar de posibles problemas.

Como vemos, el código de respuesta ante el comando **HELP** es **100**, ya que se trata de una respuesta **informativa**, y además es información general acerca de la **configuración**.

3.2.3. Listado de grupos

Lo primero que nos interesará será ver la lista de grupos que tiene el servidor. Para ello ejecutamos el comando:

LIST

A lo cual el servidor nos responderá con tropecientas líneas, cada una de las cuales tendrá el formato:
grupo último primero publicar

Donde **grupo** es el nombre del grupo al que se refiere esa línea, **último** es el número asignado dentro de ese grupo al último artículo publicado, **primero** es el número asignado al primer artículo publicado (y que, por supuesto, aún no ha caducado y se conserva en la base de datos del servidor), y **publicar** indica si en ese grupo está permitido publicar artículos (**y**), o si es de sólo lectura (**n**).

Si **último** es menor que **primero** (lo cual sería absurdo) significa que no hay ningún artículo en ese grupo.

3.2.4. Un poco de sadomaso

Si sois de los que os va la sumisión, podréis materializar vuestras fantasías a través del protocolo NNTP, ya que con el comando:

SLAVE

Informáis al servidor NNTP de que quien se conecta no es un usuario normal, ni otro servidor NNTP normal, si no un **servidor NNTP esclavo**.

Ya expliqué anteriormente en qué consiste un servidor esclavo así que, como podéis imaginar, puede tener sus ventajas el identificarse como esclavo, ya que el servidor podría darte prioridad. Por supuesto, no es muy honrado hacerse pasar por servidor esclavo sólo por conseguir un poco de prioridad, aunque eso lo dejo a la conciencia de cada uno.

Si el servidor acepta la conexión de servidores esclavos, responderá con algo como esto:

202 slave status noted

No he hecho pruebas suficientes como para hacer una estadística, pero sospecho que muchos servidores pasarán del tema, por lo que no sé hasta que punto puede ser útil este comando.

3.2.5. ¿Hay grupos nuevos?

Tal y como hemos visto, el primer paso para la distribución de grupos y artículos entre los servidores consiste en preguntar a otro servidor si ha incluido recientemente grupos nuevos en su lista.

Por supuesto, eso de "recientemente" es un término muy subjetivo, y a las máquinas no les va mucho eso de la subjetividad. Por tanto, al preguntar a un servidor si tiene grupos nuevos tenemos que indicarle también cómo de nuevos han de ser para que nos interesen (es decir, para que nos parezcan recientes). Normalmente, indicaremos como parámetro el momento (fecha y hora) en que hicimos la última consulta del mismo tipo.

Éste es un ejemplo del comando que pregunta por los nuevos grupos:

NEWGROUPS 040110 133005

El **primer parámetro** es la **fecha**, en el formato **YYMMDD**, es decir, los dos primeros dígitos identifican el **año** (¿os suena el famoso problema del año 2000 por representar años con sólo 2 cifras?...), los dos siguientes el **mes**, y los dos últimos el **día** del mes.

En el ejemplo, la fecha especificada es el 10 de Enero del año 2004.

El **segundo parámetro** es la **hora**, en el formato **HHMMSS**, es decir, los dos primeros dígitos identifican la **hora** (en formato 24h), los dos siguientes el **minuto**, y los dos últimos los **segundos**. En el ejemplo, la hora especificada es la una y media del mediodía, en el segundo 5 (13:30'05").

El problema al usar este formato es que el servidor podría estar en una franja horaria distinta a la nuestra (por ejemplo, si conectamos desde España con un servidor de USA). Si queremos garantizar que la referencia horaria es la misma, podemos añadir el parámetro **GMT**, que indica que la fecha y hora dadas son con respecto al meridiano 0:

NEWGROUPS 040110 133005 GMT

Ante este comando, el servidor siempre nos responderá con un código **231**, tanto si hay grupos nuevos como si no.

En general, en NNTP se suele indicar el final de cualquier texto mediante una línea que contenga únicamente un punto. Por tanto, si recibimos una respuesta como esta:

231 New newsgroups follow.
.

Significa que en el servidor no se ha creado ningún grupo nuevo desde la fecha y hora especificados.

Si recordamos lo que conté acerca de los códigos de respuesta, comprobamos que tiene sentido que se asigne el código **231** a esta respuesta, ya que las respuestas **2xx** corresponden a comandos ejecutados con **éxito**, y las respuestas **x3x** se refieren a cuestiones acerca del mecanismo de **distribución** de artículos y grupos entre servidores y, por supuesto, el comando **NEWGROUPS** es uno de los utilizados en la distribución de grupos.

3.2.6. ¿Hay artículos nuevos?

Como vimos, una vez que ve qué grupos nuevos ha creado el servidor, cualquier servidor que quiera continuar la distribución a través de otro servidor solicitará a continuación la lista de artículos nuevos para aquellos grupos que le interesen.

El formato del comando que permite esto es parecido al del comando **NEWGROUPS**:

NEWNEWS alt.sex.aliens 040110 133005 GMT

Con esto solicitamos al servidor la lista de artículos nuevos en el grupo **alt.sex.aliens** publicados desde las 13:30'05" GMT del 10 de Enero del 2004.

Esto sería poco eficiente si tenemos que solicitar de una en una las listas para todos los canales, por lo que podemos utilizar como comodín el asterisco (*):

NEWNEWS alt.sex.* 040110 133005 GMT

Este comando nos dará la lista de artículos nuevos para todos los grupos que pertenezcan a la categoría **alt.sex.*.**

El servidor responderá al comando NEWNEWS con el código **230**, devolviendo una línea por cada mensaje nuevo. En cada línea habrá una cadena extraña metida dentro de los caracteres **< >**.

Esta cadena es lo que se llama **Message-ID** (identificador de mensaje), e identifica unívocamente un mensaje dentro de toda la red Usenet.

Más adelante veremos para qué nos pueden servir los Message-ID.

3.2.7. Sile sile, nole nole...

Si queremos enseñar nuestros cromos al servidor, es decir, decirle qué mensajes tenemos nosotros que potencialmente puedan interesar al servidor porque no los tenga (en teoría esto ocurriría sólo si nosotros fuésemos otro servidor NNTP), utilizaremos el comando:

IHAVE<132509143@u13.hackxcrack.com>

Ese parámetro tan raro que pasamos al comando **IHAVE** es precisamente un **Message-ID.**

Es imposible saber lo que representa exactamente un Message-ID construido por otro servidor, ya que cada uno utilizará su propia heurística para identificar unívocamente sus propios mensajes. Por ejemplo, puede crear una cadena que combine la fecha y hora de creación del mensaje, junto con un identificador del usuario que lo escribió.

La heurística de identificación propia de cada servidor irá siempre antes de la **@**, ya que después de esta vendrá siempre el propio nombre de la máquina dentro de su dominio.

Tras un comando **IHAVE**, el servidor juzgará si le interesa o no ese mensaje.

Si le interesa, responderá con un código **335**, tras lo cual tendremos que enviar el cuerpo del mensaje (según el formato que veremos en el punto 3.3), terminando con una línea que contenga un único punto.

No lo he probado, pero se me ocurre que el comando **IHAVE** podría servir para sabotear los artículos publicados por otros usuarios. Si conocemos el Message-ID del artículo que queremos sabotear, y sabemos que el artículo aún no ha llegado a todos los servidores, podríamos conectarnos a los servidores que aún no tienen el artículo, decirles que tenemos un artículo con ese Message-ID, enviar un artículo falso y así, cuando llegue el auténtico artículo, el servidor creerá que ya lo tiene y, por tanto, lo rechazará.

Probablemente, esto será difícil de conseguir, ya que un servidor NNTP bien configurado debería conocer bien las IPs de los servidores NNTP conectados a él, y sólo permitir la ejecución de estos comandos a los que sabe con certeza que son servidores.

3.2.8. Solicitando un artículo nuevo

Cuando nos interese algún artículo de la lista que nos proporcionó el servidor, lo solicitaremos con el comando:

ARTICLE <27358342@n1.news.lco.es>

Donde el parámetro es el **Message-ID** correspondiente a ese artículo.

Con esto terminamos los comandos que describen la interacción típica entre dos servidores NNTP para la distribución de grupos y artículos. A continuación veremos otros comandos (o variaciones de los ya vistos), que son los que suele utilizar un usuario normal al conectar con un servidor NNTP.

3.2.9. Seleccionando un grupo

Lo más habitual es que un usuario trabaje en cada momento con un único grupo. Por ejemplo, lo típico será visualizar seguidos todos los mensajes nuevos de un mismo grupo que nos interesen, luego quizá publicar un artículo en ese mismo grupo, etc. Para indicar al servidor el grupo con el que vas a trabajar:

GROUP alt.sex.aliens

A lo cual el servidor nos responderá con un código **411** si el grupo no existe, o bien con algo como esto:

211 20 3 21 alt.sex.aliens

Donde el código **211** nos indica que el comando ha sido ejecutado con éxito, y a partir de ahora nuestro grupo de trabajo será **alt.sex.aliens** (indicado como último parámetro). Además, nos da como primer parámetro el **número total de artículos** en ese grupo (20), el número correspondiente al **primer artículo** que hay en el servidor para ese grupo (3), y el número del **último artículo** (21).

3.2.10. ¿Cómo ver un artículo?

Si no somos un servidor, si no un usuario normal, probablemente no habremos utilizado el comando **NEWNEWS**, por lo que no conoceremos el Message-ID correspondiente a cada mensaje del grupo que estamos leyendo. Por tanto, no podríamos utilizar el comando **ARTICLE** en la forma que lo hemos visto.

Por suerte, el comando **ARTICLE** permite que se le pase como parámetro, en lugar de un Message-ID, el número asignado a ese artículo dentro del grupo. Por ejemplo, para ver el mensaje 3 del grupo que acabamos de seleccionar con el comando **GROUP** (**alt.sex.aliens**), haremos:

ARTICLE 3

El servidor podrá darnos cualquiera de estas respuestas:

- **220** – Nos da el **Message-ID** del artículo para futuras referencias y, a continuación, nos transmite el **mensaje completo**.
- **221** - Nos da el **Message-ID** pero, a continuación, sólo nos transmite **la cabecera del mensaje**. El cuerpo del mensaje podremos solicitarlo a continuación, tal y como explicaré más tarde.
- **222** – Nos da el **Message-ID** pero, a continuación, sólo nos transmite **el cuerpo del mensaje** sin cabecera. La cabecera podremos solicitarla a continuación, tal y como explicaré más tarde.
- **223** – Sólo nos da el **Message-ID**. Si, a continuación, queremos que nos transmita el mensaje completo, podremos utilizar el comando **ARTICLE <message-id>** o, si sólo nos interesa una de las dos partes del mensaje (cabecera o cuerpo) podremos utilizar los comandos que explicaré más tarde.
- **412** – Indica que **no hay ningún grupo seleccionado**, debido a que no utilizamos

previamente un comando **GROUP**, por lo que el servidor no sabe a qué artículo nos estamos refiriendo.
- **423** – Nos dice **que no existe artículo** asignado al número especificado como parámetro (3) **en el grupo** seleccionado.
- **430** – Nos dice que **no existe ese artículo**.

3.2.11. ¿Cómo ver la cabecera de un artículo?

El comando **HEAD** tiene exactamente el mismo formato que el comando **ARTICLE**, pero sólo nos muestra la cabecera de un mensaje, y no el mensaje completo. Así:

HEAD <Message-ID>

Nos mostrará la cabecera de un determinado mensaje. Mientras que:

GROUP alt.sex.aliens
HEAD 3

Nos mostrará la cabecera del tercer mensaje del grupo **alt.sex.aliens**.

3.2.12. ¿Cómo ver el cuerpo de un mensaje?

Lo mismo ocurre con el comando **BODY** que, en lugar de la cabecera, nos muestra sólo el cuerpo del artículo. Ya veremos en el punto 3.3 el formato de los artículos en detalle.

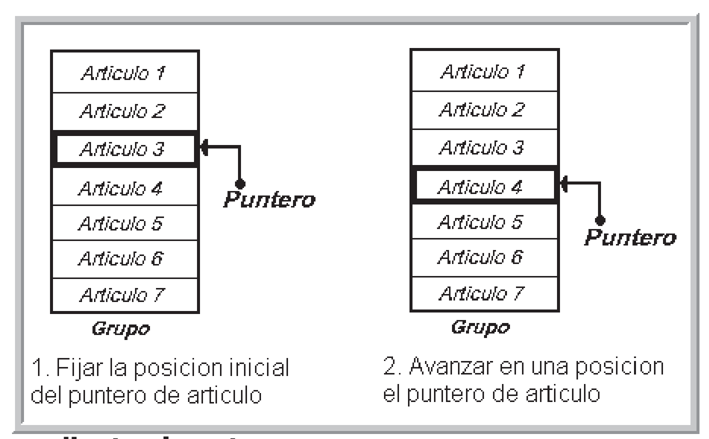
3.2.13. El puntero de artículo

Todo este sistema para recorrer los artículos en realidad no es muy práctico en muchas ocasiones.

Por ejemplo, cuando hace varios días que no visitamos Usenet y conectamos con nuestro cliente de news para descargar todos los artículos que hayan sido publicados en nuestro grupo favorito esos días, lo que sin duda hará internamente nuestro cliente es solicitar al servidor las cabeceras de todos los artículos secuencialmente. Al tener las cabeceras (con el comando **HEAD**), podrá mostrarnos de un vistazo algunos datos básicos, como el asunto o el remitente del mensaje. Una vez que tenemos la lista de cabeceras simplificadas ya podemos escoger aquellos artículos que queremos ir leyendo.

Para facilitar esta lectura secuencial de artículos, existen una serie de comandos (y variaciones de los que hemos visto hasta ahora) que permiten moverse por los artículos de forma relativa, y no absoluta. Es decir, podemos fijar un puntero que señale a un artículo en concreto, y a partir de ahí ir diciéndole tan sólo al servidor: ¡túlame el siguiente!

Fig 6.- Recorriendo una lista de artículos



mediante el puntero.

Vamos a ver cómo se consigue todo esto en los siguientes puntos.

3.2.14. Fijando la posición inicial del puntero

Lo primero que habrá que hacer para recorrer una lista de artículos mediante un puntero será fijar el puntero en su posición inicial. Hay varias formas de hacer esto. Por ejemplo,

cuando utilizamos el comando **ARTICLE**, no sólo estamos visualizando un artículo, si no que además estamos inicializando el puntero para que apunte a ese artículo. El comando cuya única función es fijar el puntero es:

STAT 3

Donde el parámetro es el **número de artículo dentro del grupo**. Por tanto, antes de un **STAT** siempre tendremos que haber ejecutado un **GROUP**, tanto para seleccionar el grupo, como para saber cuál es el primer artículo del grupo.

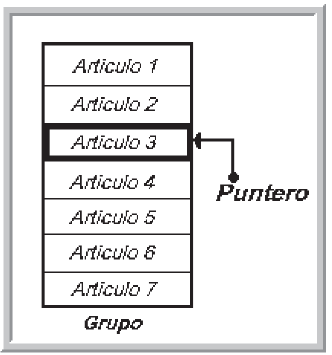


Fig 7.- Estado del puntero tras ejecutar STAT 3.

¡Ojo! Que hay que tener dos cosas en cuenta. En primer lugar, que el comando **STAT** sólo modifica el puntero de artículo si se usa como parámetro el número de artículo dentro del grupo, tal y como hemos hecho aquí, pero no si se utiliza como parámetro un Message-ID (tal y como dice en el **RFC**, no sirve para nada usar **STAT** con Message-ID, aunque no debería dar ningún error). En segundo lugar, hay que tener siempre en mente que los comandos **ARTICLE**, **HEAD**, y **BODY** también modifican el puntero de artículo, siempre que sean usados con números de artículo en lugar de con Message-ID. Por tanto, si queremos tener controlado el puntero de artículo, tendremos que tener mucho ojo a la hora de usar estos comandos.

Ahora que tenemos el puntero fijado en el artículo 3, ¿cómo podemos ver el artículo sin modificar el puntero y sin tener que usar un Message-ID (en cuyo caso todo esto no tendría ningún sentido)? Pues utilizando la última variante de los versátiles comandos **ARTICLE**, **HEAD**, y **BODY**:

ARTICLE

Escribiendo **ARTICLE** a secas visualizamos el artículo que esté en estos momentos señalado por el puntero.

HEAD

Con esto visualizaríamos sólo la cabecera del artículo que esté en estos momentos señalado por el puntero. Os dejo como gran ejercicio mental que deduzcáis que hará un comando **BODY** si no le pasas ningún parámetro. ;-)

3.2.15. ¡Siguiente!

Pos ale, ya hemos visto el primer artículo, y queremos ver el siguiente de la lista:

NEXT

Así de sencillo. Con esto hemos fijado el puntero en la siguiente posición dentro de la lista de artículos del grupo en el que estamos trabajando.

Si estábamos ya en el último artículo y no existe un siguiente, el servidor nos responderá con un error **421**. Si no, nos responderá con un código **223**, indicándonos además el Message-ID del artículo al que apunta ahora el puntero.

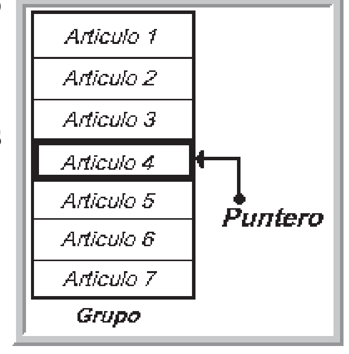


Fig 8.- Estado del puntero después de ejecutar un NEXT.

3.2.16. Vuelta atrás

También tenemos la posibilidad de avanzar hacia atrás en lugar de hacia alante:

LAST

En este caso la respuesta de error será el código **422** en caso de que estuviésemos ya en el primer artículo y no hubiese uno previo.

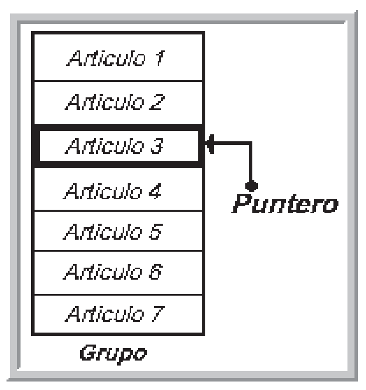


Fig 9.- Estado del puntero después de ejecutar un LAST.

3.2.17. Publicando un artículo

Para el final hemos dejado la que es sin duda una de las funciones más importantes, que es la publicación de nuestros propios artículos.

Hay que tener en cuenta antes de nada que puede haber varios motivos por los cuales no nos esté permitido publicar artículos.

En primer lugar, al conectar con el servidor éste nos tuvo que saludar con un código **200**, porque si lo hizo con un código **201** ya sabemos que podremos hacer cualquier cosa excepto publicar nuestros propios artículos.

En segundo lugar, al ver la lista de grupos (comando **LIST**) el último parámetro que nos mostraba el servidor para cada grupo nos indicaba precisamente si tenemos permiso para publicar en ese grupo (**y**) o no (**n**).

En tercer lugar, el servidor podría rechazar nuestros artículos por otros motivos justo en el momento en que nos disponemos a publicarlo.

La única forma de comprobar esto es precisamente intentando publicarlo:

POST

Con esto indicamos al servidor que queremos publicar un artículo en el grupo con el que estamos trabajando (el cual indicamos previamente con un comando **GROUP**), aunque en la cabecera del artículo será donde indiquemos finalmente sobre qué grupo o grupos queremos que se publique el artículo, como veremos más adelante.

Si al servidor le mola, nos responderá con un código **340**, invitándonos a que enviemos a continuación el artículo (recordemos que los códigos **3xx** nos informan de que un comando ha sido ejecutado con éxito, y el servidor espera por tanto a que se complete un proceso con el próximo paso).

Si al servidor no le mola que publiquemos artículos en ese grupo, nos responderá en cambio con un código **440**, y ya no habrá más que hacer.

Por tanto, después del **POST**, si recibimos un código **340** simplemente tenemos que enviar el artículo completo, con cabecera y cuerpo, siguiendo el formato que explicaré ahora mismito. :-)

3.2.18. ¡Adiós!

El último comando que falta por contar es el:

QUIT

Para los que no tengáis demasiada imaginación os explico que es el comando que sirve para terminar la sesión y cerrar la conexión con el servidor. :-P

3.3. Formato de los artículos

Como hemos ido viendo a lo largo del texto, un artículo se compone básicamente de dos partes: **cabecera** y **cuerpo**. No es ésta la única coincidencia con los mensajes de correo electrónico (tal y como vimos en los artículos sobre **POP3** y, sobre todo, sobre **SMTP**), si no que además las cabeceras guardan también muchas similitudes.

La especificación completa del formato de los artículos de Usenet se encuentra en el **RFC 1036** (aunque el **RFC 977** se refiere constantemente al **RFC 850**, en realidad el **RFC 1036** ha sustituido a este).

3.3.1. Cabecera del artículo

Las líneas de cabecera dan una serie de datos imprescindibles, o no, acerca del artículo. Existen una serie de **líneas de cabecera que son obligatorias** en todos los artículos, y son: **From, Date, Newsgroups, Subject, Message-ID, y Path**.

Como **cabeceras opcionales** tenemos: **Followup-To, Expires, Reply-To, Sender, References, Control, Distribution, Keywords, Summary, Approved, Lines, Xref, y Organization.**

Explicaré a continuación cada una de las líneas de cabecera obligatorias, y algunas de las opcionales.

3.3.1.1. From:

Especifica una dirección de correo electrónico que es (supuestamente) la del remitente. Permite incluir también el nombre completo del remitente. Podemos usar uno de estos 3 formatos admitidos:

From: PyC@LCo.es
From: PyC@LCo.es (Pepito Lopez)
From: Pepito Lopez <PyC@LCo.es>

El primer formato especifica sólo la dirección de correo, mientras que los otros dos especifican además el nombre completo.

Si tenéis imaginación y habéis seguido el resto de la serie RAW, probablemente se os habrá ocurrido probar a poner una dirección de correo falsa en el **From**. ;-)

3.3.1.2. Date:

Al igual que en el caso del correo electrónico, las cabeceras de NNTP siguen el estándar **MIME**, así que hay que utilizar un formato de fechas admitido por este estándar. Si, por ejemplo, queremos especificar el 21 de Noviembre del año 2003, a las 13:30'05" según el meridiano 0 (GMT):

Date: Fri, 21 Nov 03 13:30:05 GMT

3.3.1.3. Newsgroups:

Indica el grupo de destino en el que queremos publicar el mensaje:

Newsgroups: alt.sex.aliens

Si queremos iniciar un **crossposting**, enviando el mensaje a varios grupos, podemos separarlos por comas:

Newsgroups: alt.sex.aliens,alt.sex.ganimedes,alt.sex.kripton

3.3.1.4. Subject:

Al igual que en un mensaje de correo electrónico, es un texto que especifica el asunto del mensaje. En caso de que sea respuesta a otro mensaje debería empezar con el prefijo **"Re:",** aunque no es obligatorio. Lo que si que será obligatorio en este caso es incluir la línea de cabecera opcional **References**, tal y como veremos más adelante.

3.3.1.5. Message-ID:

Ya hemos visto antes en qué consistía un Message-ID. Siempre hemos de encuadrarlos entre los caracteres **<** y **>**.

3.3.1.6. Path:

Esta línea es bastante interesante, ya que ayuda mucho a mejorar el rendimiento en la distribución de artículos. Cada vez que un artículo pasa por un servidor, éste añade su nombre en esta línea. Así, cuando un servidor

A solicite la lista de artículos nuevos a un servidor **B**, y éste vea que en el **Path** de un artículo se encuentra el nombre del servidor **A**, el servidor **B** sabrá entonces que **A** conoce ya ese artículo, por lo que se ahorrará incluir ese artículo en la lista que facilitará al servidor **A**.

En el **Path** cada servidor va añadiendo su nombre, separándolo de los demás con un signo "**!**":

Path: news.hackxcrack.com!n133.Lco.es!@nss.mad.ttd.es

Si el servidor **news.hackxcrack.com** solicitase artículos nuevos a otro servidor, éste no incluiría en su lista de artículos nuevos aquellos que tuviesen un **Path** como este, en el que se encuentra ya el servidor **news.hackxcrack.com**.

Un servidor malicioso podría sabotear la distribución de artículos hacia un servidor si incluyese en el **Path** de los artículos que recibe la dirección del servidor al que quiere sabotear, aparte de la suya propia.

3.3.1.7. Followup-To:

Para comentar algunas de las cabeceras opcionales (el artículo sería demasiado extenso si comentase todas), empezaré por esta, ya que es interesante por su relación con algunos aspectos que hemos visto a lo largo del artículo. En concreto, puede ser útil para evitar el **crossposting**.

Si tenemos, por ejemplo, un artículo que incluye estas dos líneas de cabecera:

Newsgroups: alt.sex.aliens,alt.sex.ganimedes,alt.sex.kripton
Followup-To: alt.sex.aliens

Este artículo se habrá publicado en los grupos **alt.sex.aliens, alt.sex.ganimedes**, y **alt.sex.kripton** pero, en cambio, las respuestas a ese artículo serán publicadas sólo en el grupo especificado en la línea **Followup-To**, es decir, en **alt.sex.aliens**.

Esta es la forma correcta de publicar un artículo cuando es realmente necesario hacerlo en varios grupos, ya que no se generará un crossposting, pues todas las respuestas quedarán en un único grupo.

Por supuesto, en los artículos construidos de este modo es conveniente indicar en el texto del mensaje que aquellas personas que quieran seguir el hilo de este artículo deberán hacerlo en el grupo **alt.sex.aliens**, ya que la mayoría de la gente no suele ojear las cabeceras antes de responder a un artículo. ;-)

3.3.1.8. References:

Cuando el artículo sea respuesta a otro artículo, deberá estar presente esta línea, que contendrá como parámetro el Message-ID del artículo original del cual es respuesta.

3.3.1.9. Control:

Esta línea de cabecera se usa para una serie de "artículos" especiales que contienen comandos de control para los servidores NNTP.

Los mensajes de control permiten varias funciones, como cancelar la distribución de un artículo previamente publicado, avisar a otro servidor de la creación o destrucción de un grupo, etc.

Como ya dije cuando hablé del mecanismo de distribución de grupos y artículos, es más eficiente avisar de la creación de grupos nuevos según van apareciendo, en lugar de esperar a que nos pregunten por ellos. La forma de

conseguirlo es mediante los comandos de control.

La especificación detallada de estos comandos de control la podéis encontrar en **el RFC 1036**, ya que el espacio para el artículo se me está acabando ya. :-(

3.3.2. Cuerpo del artículo

Por último, acerca el cuerpo del artículo sólo he de mencionar un par de cosas.
En primer lugar, como en NNTP para indicar el fin de un texto se utiliza **una línea con un único punto**, si queremos, por algún extraño motivo, que nuestro texto incluya una línea con un punto, tendremos que poner **dos puntos** en lugar de uno. Es decir, un texto como este:



Será interpretado como:



(Se supone que es una cara). Ya que siempre que hay dos puntos al comienzo de una línea se contraen en uno solo, para distinguirlo del punto simple que finalizaría el cuerpo del mensaje.

Como último comentario, si queremos incluir **archivos binarios** tendremos que documentarnos más, ya que hay que codificarlos mediante **uuencode** siguiendo el estándar **MIME**, y todo esto se sale con mucho de lo que puede abarcar el artículo. :-(

4. RESUMIENDO

Creo que es necesario hacer una sesión completa de ejemplo, porque ahora mismo debéis de tener miles de ideas flotando en la cabeza, pero nada concreto.

Así que aquí os pongo una sesión completa de ejemplo, y os voy comentando cada paso (los comentarios irán en color negro). El servidor al que nos conectaremos será **nsnmrro2-lo.nuria.telefonica-data.net**

Os dejo ya con esto, y espero veros el próximo mes no sólo con un artículo, si no con dos! :-O ¿Tenéis curiosidad? Pues esperad al próximo mes, para ver de qué estoy hablando. ;-)

telnet nsnmrro2-lo.nuria.telefonica-data.net 119 (Nos conectamos al servidor)

200 nsnmrro2-lo.nuria.telefonica-data.net InterNetNews NNRP server INN 2.3.2 ready (posting ok). El servidor nos saluda con código 200, por lo que podemos utilizar este servidor para publicar artículos.

GROUP alt.sex.aliens Seleccionamos este grupo para trabajar con él

211 77 14766 14847 alt.sex.aliens El servidor nos informa de que el grupo seleccionado será a partir de ahora nuestro grupo de trabajo. En el servidor hay 77 mensajes almacenados en ese grupo, que van desde el número 14766 hasta el número 14847,

HEAD 14840 Queremos ver la cabecera del artículo 14840 del grupo alt.sex.aliens.

221 14840 <607b02db.0401071042.3f62ccf4@posting.google.com> head La primera línea de la respuesta al comando HEAD nos dice que nos va a mostrar lo que pedimos (código 211), que el número del artículo que va a mostrar es el 14840, y nos dice además el Message-ID de ese artículo.

Path: nsnmrro2-lo.nuria.telefonica-data.net!nsnmrro1-lo.nuria.telefonica-data.ne t!in.100proofnews.com!in.100proofnew

s.com!hermes.visi.com!news-out.visi.com!petbe.visi.com!newsfeed2.dallas1.level3.net!news.level3.com!postnews1.google.com!not-for-mail En esta línea vemos los servidores por los que ha circulado este artículo (al menos hasta donde sabe éste servidor), para evitar que sea distribuido una y otra vez a los mismos servidores.

From: gaxa7a5@yahoo.com (gaxa7a5) Aquí tenemos el remitente del artículo.

Newsgroups: alt.sex.algore.lick.the.wet.toad,vegas.personals.fsm,alt.sex.algore.little.wooden.boy,alt.sex.aliens,alt.sex.alt.syntax.tactical Como vemos, este artículo fue enviado a varios grupos, y no sólo a éste. Esto ya va dando que pensar bastante... A ver si va a tratarse de spam...

Subject: HOW I GOT A 9 INCHES PENIS... READ MY STORY! / 01.07.2004 10:48 / Creo que por el asunto del mensaje queda ya patente que se trata de spam, a no ser que... ¿y si contactando con algún alien ha conseguido este hombre aumentar su virilidad y quiere compartir su experiencia con el resto de la comunidad?

Date: 7 Jan 2004 10:42:34 –0800 Fecha de publicación según el formato MIME.

Organization: http://groups.google.com Línea de cabecera opcional que normalmente da información acerca de la organización a la que pertenece el remitente. En este caso, ha sido Google quien ha añadido la línea, indicando que el artículo fue enviado a través de Google.

Lines: 17 Otra línea opcional, que indica el número de líneas que hay en el cuerpo del artículo.

Message-ID: <607b02db.0401071042.3f62ccf4@posting.google.com> Esto ya no es opcional, ya que es el identificador de mensaje de este artículo. Vemos de nuevo que el artículo ha sido enviado a través de Google.

NNTP-Posting-Host: 209.115.59.183 Otra línea opcional que nos da la IP desde la que se publicó el artículo. :-O

Content-Type: text/plain; charset=ISO-8859-1 ¡Horror! Ya estamos otra vez con el MIME. ¿Acaso no lo sospechábais? Como ya os dije, los artículos de Usenet siguen el estándar MIME, por lo que las líneas de cabecera opcionales incluyen muchas de las líneas del estándar MIME. Os recuerdo que esta línea informa acerca del formato del texto que se encuentra en el cuerpo del artículo.

Content-Transfer-Encoding: 8bit Cuando se trate de texto puro, a veces lo podremos encontrar codificado en sólo 7 bits.

X-Trace: posting.google.com 1073500955 17623 127.0.0.1 (7 Jan 2004 18:42:35 GMT) Línea opcional que permite seguir el rastro a la publicación del artículo. En este caso la cosa queda bastante clara con las líneas anteriores.

X-Complaints-To: groups-abuse@google.com Mira tu por donde, que esta línea nos puede ser útil, ya que nos informa de dónde debemos acudir en caso de que detectemos que el remitente ha abusado de los servicios ofrecidos por Google. Como en este caso se trata de spam, si tuviésemos mala leche podríamos notificarlo en esta dirección aunque, sinceramente, dudo que sirviese para nada. :-)

NNTP-Posting-Date: Wed, 7 Jan 2004 18:42:35 +0000 (UTC) Aquí me han pillado... ¿para qué servirá esta línea opcional, si la línea

obligatoria Date ya informa de la fecha de publicación? :-m

Xref: nsnmrro2-lo.nuria.telefonica-data.net alt.sex.algore.lick.the.wet.toad:3805 vegas.personals.fsm:235 alt.sex.algore.little.wooden.boy:8075 alt.sex.aliens:1 4840 alt.sex.alt.syntax.tactical:13133 En primer lugar nos da la dirección del servidor al que estamos conectados, para luego decirnos el estado de este artículo en la base de datos de éste servidor. Ya sabemos que el artículo es el 14840 del grupo alt.sex.aliens, pero esta línea nos dice que este artículo se encuentra también en otros grupos (como ya sabemos, gracias a la línea Newsgroups), y nos dice el número de artículo que ocupa dentro de cada uno de esos otros grupos. Esto permitirá a nuestro cliente identificar este artículo aunque estemos en otros grupos, para marcarlo como ya leído y evitarnos releer el mismo spam una y otra vez en todos los grupos. Por ejemplo, este artículo es el número 3805 del grupo alt.sex.algore.lick.the.wet.toad.

. Esta línea que contiene sólo un punto finaliza la transmisión de la cabecera que habíamos solicitado. Si en lugar de HEAD hubiéramos utilizado el comando ARTICLE, aquí habría una línea en blanco, después el cuerpo del mensaje y, por último, una línea como ésta que sólo tuviese un punto.

NEXT Movemos el puntero al siguiente artículo, es decir, al 14841.

223 14841 <7a196ddb.0401071956.4ce763e4@posting.google.com> Article retrieved; request text separately. El servidor nos dice que el puntero se sitúa ahora en el artículo 14841 del grupo en el que estamos, tal y como habíamos pedido. Nos da, además el Message-ID del artículo.

BODY Esta vez vamos a pedir el cuerpo del artículo 14841.

222 14841 <7a196ddb.0401071956.4ce763e4@posting.google.com> body Nos da el Message-ID del artículo, para mostrarnos su cuerpo a continuación.

FAT DUMPY EXWIFE FOUND TAKING DUSTBATH ON MARS. AMAZIINGLY THE PHOTO ONLY SHOWED PORTIOS BUT IT WAS VERY FIGHTENING TO SEE. El "artículo" nos cuenta la apasionante historia de una gorda haciendo cochinadas en Marte.

. La línea con el punto termina el texto.

QUIT Con lo de la gorda en Marte ya hemos tenido suficiente por hoy...

205 . Usease: ¡Hasta luego!

Autor: PyC (LCo)
Ilustraciones: Adhara (LCo)



CONSIGUE LOS NUMEROS ATRASADOS EN:

WWW.HACKXCRACK.COM



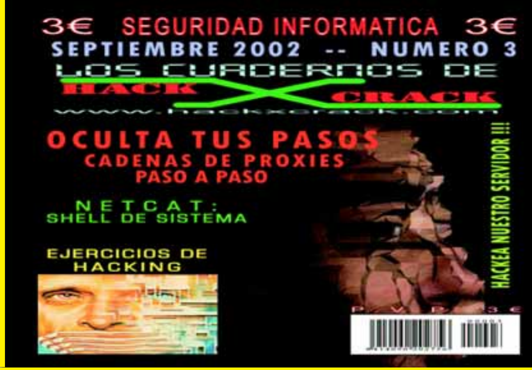
NÚMERO1:

-CREA TU PRIMER TROYANO
 INDETECTABLE POR LOS ANTIVIRUS.
-FLASHFXP: SIN LÍMITE DE VELOCIDAD.
-FTP SIN SECRETOS: PASVMODE.
-PORT MODE/PASV MODE Y LOS FIREWALL: LA UTILIDAD DE LO APRENDIDO.
-TCP-IP:INICIACIÓN (PARTE 1).
-EL MEJOR GRUPO DE SERVIDORES FTP DE HABLA HISPANA.
-EDONKEY 2000 Y SPANISHARE.
-LA FLECHA ÁCIDA.



NÚMERO 2:

-CODE/DECODE BUG: INTRODUCCIÓN.
-CODE/DECODE BUG: LOCALIZACIÓN DEL OBJETIVO.
-CODE/DECODE BUG: LÍNEA DE COMANDOS.
-CODE/DECODE BUG: SUBIENDO ARCHIVOS AL SERVIDOR REMOTO.
-OCULTACIÓN DE IP: PRIMEROS PASOS.
-LA FLECHA ÁCIDA: LA SS DIGITAL.
 AZNAR AL FRENTE DE LA SS DEL SIGLO XXI.



NÚMERO 3:

-PROXY: OCULTANDO NUESTRA IP.
 ASUMIENDO CONCEPTOS.
-PROXY: OCULTANDO NUESTRA IP.
 ENCADENANDO PROXIES.
-PROXY: OCULTANDO NUESTRA IP.
 OCULTANDO TODOS NUESTROS PROGRAMAS TRAS LAS CADENAS DE PROXIES.
-EL SERVIDOR DE HACKXCRACK: CONFIGURACIÓN Y MODO DE EMPLEO.
-SALA DE PRACTICAS: EXPLICACIÓN.
-PRÁCTICA 1ª: SUBIENDO UN ARCHIVO A NUESTRO SERVIDOR.
-PRÁCTICA 2ª: MONTANDO UN DUMP CON EL SERV-U.
-PRÁCTICA 3ª: CODE/DECODE BUG. LÍNEA DE COMANDOS.
-PREGUNTAS Y DUDAS.



NÚMERO 4:

- CREA TU SEGUNDO TROYANO, INDETECTABLE E INMUNE A LOS ANTIVIRUS.
 CONOCIENDO EL RADMIN.
 GESTIONANDO UNA SALA DE ORDENADORES.
 OCULTANDO EL RADMIN.
 INSTALANDO EL RADMIN EN EQUIPOS REMOTOS.
- OCULTACIÓN DE IP POR NOMBRE DE DOMINIO.
- CREA LETRAS DE IMPACTO PARA TUS DOCUMENTOS (LETRAS DE FUEGO).
- CONSIGUE UNA IP FIJA.



NÚMERO 5:

-HACK-OPINION: LA PIRATERÍA EN INTERNET.
-ROOTKITS: LA PESADILLA DE CUALQUIER ADMINISTRADOR.
-ROOTKITS: EL SR. NTROOT.
-WAREZ: APPZ, GAMEZ, MP3Z, DIVX, FTPZ, 0-DAY.
-APRENDIENDO A COMPILAR PROGRAMAS. COMPILA TU PROPIO NETCAT.
-BUGS, ERRORES Y OTRAS FORMA DE JOD...
-NETBIOS: ESTUDIO Y PENETRACIÓN DE SISTEMAS.
-ASESINADOS POR LA LSSI.
-LISTADO DE ORDENES PARA NETBIOS.
-HACK-OPINION: PAGOS POR INTERNET SEGUROS YÁ.



NÚMERO 6:

- PASA TUS PELICULAS A DIVX (STREAMING)
- PASA TUS PELICULAS A DIVX II (CODEC DIVX)
- PUERTOS & SERVICIOS
- eMule: EL NUEVO REY DEL P2P
- NUEVA SECCION: PROGRAMACION DESDE 0
- CURSO DE VISUAL BASIC
- IPHXC: EL TERCER TROYANO DE HXC
- TENDENCIAS ACTUALES EN CODIGO MALICIOSO
- OCULTACION DE FICHEROS. METODO STREAM (ads)
- TRASTEANDO CON EL HARDWARE DE UNA LAN



NÚMERO 7:
- PROTOCOLOS: POP3
- PASA TUS PELICULAS A DIVX III (EL AUDIO)
- PASA TUS PELICULAS A DIVX IV (MULTIPLEXADO)
- CURSO DE VISUAL BASIC: LA CALCULADORA
- IPHXC: EL TERCER TROYANO DE HXC II
- APACHE: UN SERVIDOR WEB EN NUESTRO PC
- CCPROXY: IV TROYANO DE PC PASO A PASO
- TRASTEANDO CON EL HARDWARE DE UNA LAN

NÚMERO 8:

- CURSO DE LINUX
- APACHE: COMPARTE ARCHIVOS
- REVERSE SHELL
- CURSO DE VISUAL BASIC: MAS CALCULADORA
- PROTOCOLOS Y SU SEGURIDAD: SMTP



NÚMERO 9:

- CURSO DE LINUX (Sistema de archivos)
- APACHE: COMPARTE ARCHIVOS MEDIANTE WEB.
- CURSO DE VISUAL BASIC: MI 1ª DLL \ ACCESO A DATOS
- PORT SCANING: NMAP
- SERIE RAW: IRC



NÚMERO 10:

- CURSO DE LINUX (Gestión de usuários)
- APACHE + MySQL + PHP=Trio de Ases
- CURSO DE VISUAL BASIC: ACCESO A DATOS(II)
- XML: El futuro de la trasferencia de datos
- SERIE RAW: DCC



NÚMERO 11:
- Curso de linux: programacion
- Visual Basic: IIS bug exploit
- Apache como proxy
- Serie Raw: FTP
- Validacion XML: DTD
- Historia: Lady Augusta Ada Byron



NÚMERO 12:
- Curso de linux: programacion C.
- Visual Basic: IIS bug exploit. Nuestro primer Scanner.
- APACHE: Configuralo de forma segura.
- Serie Raw: FTP(II)
- VALIDACION XML: DTD (II)



NÚMERO 13:
- Curso de linux: programacion C(II).
- Visual Basic:Nuestro primer proyecto.
- APACHE: Configuralo de forma segura.
- Serie Raw: HTTP.
- CURSO XML: DOM.



NÚMERO 14:
- Curso de linux: programacion C(III).
- Visual Basic:Nuestro primer proyecto(II).
- Curso de PHP
- Serie Raw: DNS.
- CURSO XML: DOM(II).
- HIJACKING



- CURSO DE PHP (II)
- Xbox. Instalar Linux
- SERIE RAW (9): MSN
- CURSO VISUAL BASIC: UN CLIENTE, UNA NECESIDAD(III).
- PROGRAMACION BAJO LINUX: LENGUAJE C(III)

CONSIGUE LOS NUMEROS ATRASADOS EN:



Lubic

crea tus propias cajas

by **AeroCool**

www.aerocool.com.tw



IMPORTADOR OFICIAL
EXCLUSIVO ESPAÑA-PORTUGAL

TEL.: 902-227733 / 945-176647
www.biomag.biz / compras@biomag.biz

Los paquetes estandar son:

1. LUBIC-3519-BL
2. LUBIC-3519-BK
3. LUBIC-3519-SV
4. LUBIC-4480-BL
5. LUBIC-4480-BK
6. LUBIC-4480-SV
7. LUBIC-AIRPLANE-BL
8. LUBIC-AIRPLANE-BK
9. LUBIC-AIRPLANE-SV

NUEVA TECNOLOGIA DE TUBO SUPERCONDUCTOR

Serie Deep Impact
DP-102

Características

1. Alta conductividad térmica
2. Totalmente fabricado en cobre con un tubo superconductor para un maximo rendimiento
3. Super Diseño circular
4. DP-102 ofrece una solución de doble ventilador - pueden usarse uno o dos ventiladores.
5. Incluye dos sets de adaptadores de ventilador de 70mm a 80mm.
6. El disipador puede ser rotado 360 grados y los ventiladores pueden ser instalados en cualquier posición.
7. DP-102 puede ser "Universal" compatible con AMD e Intel P4

Aplicaciones

AMD: Athlon XP 3600+ y superiores
Intel: P4 Socket 478 3.6Ghz y superiores
(P4 sólo Versión Universal)

Maximo Diseño

1 ventilador

2 ventiladores

Disipador

Dimensiones =Tubo -100 mm, 36+ láminas - dia. 66mm
Material =Tubo Superconductor + Láminas de Cobre

¿Aburrido de las mismas cajas de ordenador de siempre?

¡¡Ahora, transforma tu ordenador "cuadrado" en cualquier cosa!! Presentamos las nuevas cajas "LUBIC". Gracias al concepto modular de "LUBIC", dispones de todo lo necesario para crear tu propio diseño de caja de ordenador con los elementos de aluminio incluidos en el producto

Planifica cuidadosamente el plan de construcción y los materiales para transformar tu ordenador en algo más que una simple y aburrida caja de metal.
¡¡Todo es posible con las cajas "LUBIC" !! Puedes crear una caja "Escorpión", una caja "Elefante", una caja "bombardero B-52" o cualquier cosa que imagines. También puedes crear cosas que no sean ordenadores con los modulos "LUBIC" como portaretratos, expositores, mesas...

¿Es el cielo el limite?
Con los módulos "LUBIC", ¡¡NO EXISTEN LIMITES!!!
¡¡¡Da forma a todas sus ideas!!! ¡¡Las creaciones solo estan limitadas por tu imaginación!!!...







X — For Xtremely Cool People — Suitable For Xtreme Case Modding Solutions

NUEVA TECNOLOGIA DE TUBO SUPERCONDUCTOR

Serie High Tower
HT-101

Características

1. Alta conductividad térmica
2. Totalmente fabricado en cobre con 3 tubos superconductores para un rendimiento extremo.
3. HT-101 ofrece una solución de doble ventilador - pueden usarse uno o dos ventiladores.
4. Carcasa de plástico azul sensible a la luz ultravioleta.
5. Ventilador con 4 LEDS UltraVioleta - innovadoras aspas "flower shape" para super rendimiento y silencio extremo.

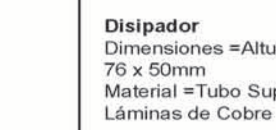
Velocidad: 2500 RPM
Caudal: 29.6 CFM
Ruido: 22 DBA

Aplicaciones

AMD: Athlon XP 3600+ y superiores
Intel: P4 Socket 478 3.6Ghz y superiores

Disipador

Dimensiones =Altura -113 mm, 31 láminas, 76 x 50mm
Material =Tubo Superconductor + Láminas de Cobre



¡¡Reactivo UV!!